UNIVERSITY OF CALIFORNIA
SANTA CRUZ

# USING DOMINATING SETS TO IMPROVE THE PERFORMANCE OF MOBILE AD HOC NETWORKS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

**Marco Aurélio Spohn**

September 2005

The Dissertation of Marco Aurélio Spohn
is approved:

_____

Professor J.J. Garcia-Luna-Aceves, Chair

_____

Professor Katia Obraczka

_____

Professor Emmet James Whitehead, Jr.

_____

Robert C. Miller
Vice Chancellor for Research and
Dean of Graduate Studies

# Report Documentation Page

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE | 2. REPORT TYPE | 3. DATES COVERED |
|---|---|---|
| **SEP 2005** | | **00-09-2005 to 00-09-2005** |

| 4. TITLE AND SUBTITLE | 5a. CONTRACT NUMBER |
|---|---|
| **Using Dominating Sets to Improve the Performance of Mobile Ad Hoc Networks** | 5b. GRANT NUMBER |
| | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | 5d. PROJECT NUMBER |
| | 5e. TASK NUMBER |
| | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064** | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSOR/MONITOR'S ACRONYM(S) |
| | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT
**Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT **unclassified** | b. ABSTRACT **unclassified** | c. THIS PAGE **unclassified** | | **164** | |

# Contents

# List of Figures

# List of Tables

# List of Notations

| | |
|---|---|
| $G = (V, E)$ | A graph, $G$, where $V$ is the set of nodes (vertices), and $E$ is the set of links (edges) |
| $N_1^i$ | The set of one-hop neighbors of node $n_i$ |
| $N_2^i$ | The set of two-hop neighbors of node $n_i$ |
| $N_i$ | The two-hop neighborhood of node $n_i$ (i.e.,$N_i = N_1^i \cup N_2^i$ ) |
| $N_r^i$ | The set of r-hop neighbors of node $i$ (assuming $N_0^i = \{n_i\}$) |
| $\mathfrak{N}_{r,i}$ | The r-hop neighborhood of node $i$; i.e., $\mathfrak{N}_{r,i} = \bigcup_{k=0}^{r} N_k^i$ |
| $\Delta$ | The largest cardinality among all $r$-hop neighborhoods in the network. That is, $\Delta = max(\lvert\mathfrak{N}_{r,k}\rvert), \forall\ k \in V$ |
| $D^j_{\text{1-hop}}$ | *one-hop dominating nodes* of node $n_j$ (computed via DP or via MPR, considering node $j$ as the source). That is, $D^j_{\text{1-hop}} \subset N_1^j$ such that $(\bigcup_{k \in D^j_{\text{1-hop}}} N_1^k) = N_2^j$ |
| $\mathcal{C}$ | List of candidates to be forwarders (used in THP) |
| $\mathcal{U}[j]$ | List of *one-hop dominating nodes* of node $n_j$ (i.e., $\mathcal{U}[j] \subset D^j_{\text{1-hop}}$) that need to be covered (used in THP) |
| $U_i$ | List of two-hop nodes that need to be covered by *Dominant Pruning* (i.e., $U_i = N_2^i - N_1^i$) |
| $F^i_{DP}$ | The forwarder list computed by node $n_i$ using *Dominant Pruning* |
| $\mathcal{F}_{\mathsf{S}}$ | The sender's forwarder list (used in EDP) |
| $\mathcal{F}_{\mathsf{S_S}}$ | The second-to-previous forwarder list (used in EDP) |
| $\mathcal{F}_i$ | The EDP or THP forwarder list as computed by node $n_i$ |

## Abstract

Using Dominating Sets to Improve the Performance of Mobile Ad Hoc Networks

by

Marco Aurélio Spohn

A *mobile ad hoc network* (MANET) is a wireless network that does not rely on any fixed infrastructure (i.e., routing facilities, such as wired networks and access points), and whose nodes must coordinate among themselves to determine connectivity and routing. Coordination in ad hoc networks includes operations such as neighborhood discovery, organization of nodes (i.e., topology control and clustering), and routing. Most mechanisms performing these operations employ broadcasting of signaling messages as the underlying mechanism. The broadcast can target a portion of the network (e.g., gathering neighborhood information), or the entire network (e.g., discovering routes on demand). The focus of this thesis is the design and analysis of algorithms that improve broadcasting and hierarchical organization in ad hoc networks. To design such algorithms, concepts from domination in graphs are explored, because of their similarities to the problems arising with the broadcasting of signaling and data in MANETs.

To my wife,

Tânia,

and my daughter Felícia.

## Acknowledgements

The friendship and love of my wife, Tânia, and the grace and love of my daughter, Felícia, were essential for the successful development of my graduate studies.

I want to express my deep gratitude to my adviser, Professor J.J. Garcia-Luna-Aceves for his continuous support, guidance, and understanding during my journey of doctoral study and research. His positive attitude, enthusiasm, good energy, and friendship are an inspiration.

I thank the members of my committee, Professors Katia Obraczka and Jim Whitehead for their valuable advice and feedback. I would also like to thank Carol Mullane for her friendly help and advice.

A big thanks to my fellow *cocos* (Marcelo Carvalho, Yu Wang, Saro, Ramesh, Ravindra, Hari, Marc, Long, Zhenjiang, Renato, Brad, Soumya, Lori and Radhika) in the *Computer Communications Research Group* (CCRG) directed by J.J., and students (Cintia, Nacho, Kumar and Venkatesh) in Katia's lab for their fellowship.

I cannot forget to thank my parents, Verno and Remi, without whom I could not have started this journey. Thanks mom and dad! A big thanks to my brother, Marcelo, for his support and friendship.

Last but not the least, this work was supported in part by CNPq (*Conselho Nacional de Desenvolvimento Científico e Tecnológico*, Brazil), the Jack Baskin Chair of Computer Engineering at UCSC, the U.S. Air Force/OSR under grant No. F49620-00-1-0330, the National Science Foundation under grant CNS-0435522, and by UCOP CLC under grant SC-05-33.

# Chapter 1

# Introduction

During the last decade, wireless networks have been an important topic of research in computer networks, with special attention focused on *mobile ad hoc networks* (MANETs). A MANET is a wireless network that does not rely on any fixed infrastructure, and whose nodes must organize themselves to determine connectivity and routing. Because of mobility and radio channel connections, the links between nodes are unreliable.

A MANET can be deployed rapidly in disaster relief (e.g., in disaster areas with scarce or non-existing communication infrastructure), battlefield communication (e.g., to provide communication among platoon members), collaborative computing (e.g., a group of persons using their laptop computers to set up a wireless network to exchange data during a meeting), and any other situation requiring a network built on demand.

In a MANET, radio waves are used for communication among nodes. The transmitter and receiver do not have to be aligned physically, because radio waves are omnidirectional (i.e., they travel in all directions). On the other hand, it is more complicated to detect if a

node's transmission collides with any other simultaneous transmission. Because of that, most *medium access* (MAC) protocols for MANETs are contention based with a collision avoidance mechanism. Depending on the location of nodes, network packets may require intermediary nodes to relay them until they reach the destination. In this case, routes can be found on demand (i.e., computed as needed), or proactively (i.e., routes are computed to all nodes regardless of any data flow). Once a route is available, intermediary nodes transmit the packet toward the next hop to the destination; eventually, the packet reaches the destination (assuming the network is connected). With some adjustments, any TCP/IP [58] based protocol is feasible on top of the routing layer of a MANET.

In MANETs, many underlying protocols (e.g., routing protocols and topology control) use some form of flooding to send control messages. Usually it is assumed that the actual position of any node is not known, because of node mobility. In this case, whenever a control message must reach any node in the network (e.g., when searching for a route to any destination), flooding is the only alternative, because restricting the broadcast to parts of the network may not cover the destination. Organizing the nodes in a hierarchical structure such as a *broadcast tree* may provide more efficient broadcast operations, because only tree members are assigned to perform broadcast transmissions during the flooding operation. In contention-based MAC protocols, reducing the number of broadcast transmissions translates into less contention, and fewer packet collisions. Hence, all protocols can benefit from improvements to the basic flooding mechanisms.

The focus of the research reported in this thesis is on developing and analyzing algorithms that improve broadcasting and hierarchical organization in ad hoc networks. To design

such algorithms, concepts from *domination in graphs* [24] are explored, because of their similarities to the problems arising with the broadcasting of signaling and data in MANETs.

## 1.1 Physical Layer and Medium Access in MANETs

In MANETs, all protocols are restricted to the capacity of wireless transmissions (i.e., radio transmissions). Due to radio waves' ability to travel long distances, interference between wireless devices is the most significant source of problems in MANETs. MAC protocols must be designed to reduce the impact of interference, and provide some guarantees of service to the above layer (i.e., network layer).

Single channel wireless networks, in which nodes share a common frequency and modulation scheme, suffer from a type of interference called *hidden terminal* [59]. Figure 1.1 shows an example. Nodes 1 and 3 are out of range of each other (i.e., they are hidden from each other), but node 2 is within range of both nodes 1 and 3. If nodes transmit with no coordination, it is possible that nodes 1 and 3 transmit concurrently causing interference at node 2, what may prevent node 2 from hearing either.

MAC schemes can be broadly classified into two categories: on-demand (contention-based) and scheduled [7]. An on-demand scheme determines the pair of communicating nodes through the exchange of control messages before each transmission. On the other hand, scheduled schemes prearrange or negotiate a set of timetables for individual nodes or links, such that the transmissions are collision-free in the time and frequency domains. The main disadvantage of this approach is that it requires node synchronization, which is hard to accomplish in MANETs, because nodes can join or leave the network at any time.

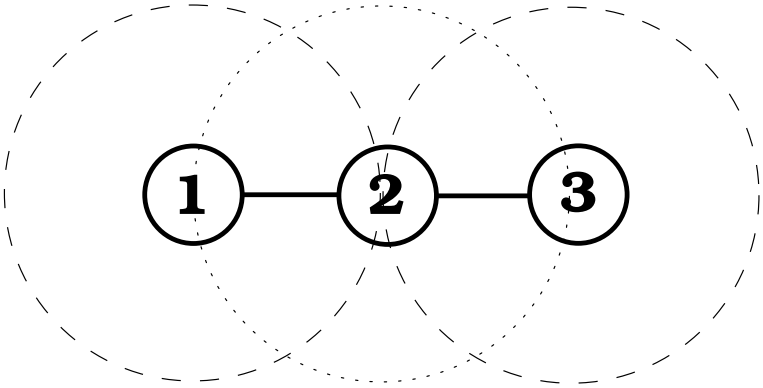



**Figure 1.1**: Hidden terminal problem

In contention based medium access protocols, the number of packet collisions increases proportionally to the number of broadcast transmissions. Increased packet collisions compromises network performance. Many operations related to the coordination of nodes incur redundant broadcast transmissions. In this case, applying mechanisms to reduce redundancy, while guaranteeing the operations' reliability, is desirable.

## 1.2 Coordination in MANETs

In a MANET, nodes must coordinate among themselves without resorting to any pre-existing network infrastructure (i.e., routing facilities, such as wired networks and access points). Broadcasting of signaling messages is the underlying mechanism for coordination, and the broadcast can target a portion of the network (e.g., gathering neighborhood information), or the entire network (e.g., discovering routes on demand).

Coordination in ad hoc networks includes operations such as neighborhood discovery, organization of nodes (i.e., topology control and clustering), and routing. Examples of

organization of nodes include the location of services, computing an efficient backbone for the broadcasting of signals, and routing of data packets (routing can benefit from some specific organization structure, but this is not a requirement for the routing of data packets).

### 1.2.1 Organization of Nodes

Organization of nodes can be static (i.e., performed proactively), or dynamic (i.e., performed on demand). While operations to build such structures require broadcasting of signaling messages, these structures make broadcast operations scale to much larger portions of the network. That is, the hierarchical structure functions as a backbone, on top of which broadcasting can be performed more efficiently.

In MANETs, there are two broad categories of hierarchical architectures: *clustering* [12], and *topology control based on hierarchies* [8, 34]. These architectures can be used to prolong the network's lifetime [6, 10, 73], attain load balancing [8], and increase network scalability [25, 34].

With *clustering* [12], the substructures that are collapsed in higher levels are called *clusters*. In each cluster, at least one node may represent the cluster, and this node is usually called a *cluster-head*. Cluster-heads act as leaders in their clusters, providing some service to their members. For example, a cluster-head could be an access point to the outside network, or it could be a *sink* for collecting information from a group of sensors (cluster members) in a sensor network [2].

Topology control based on hierarchies and clustering are closely related problems. While the former defines a real *backbone* of the network (i.e., the nodes in the backbone are

connected, covering all nodes in the network), the latter constructs a *virtual backbone* (i.e., the set of cluster-heads are not necessarily connected, but they cover all nodes in the network). An example of an operation requiring a real backbone is the flooding of control messages.

### 1.2.2 Routing

There are two broad categories of unicast routing protocols for MANETs, proactive and reactive. With *proactive routing* (e.g., OLSR [14]), nodes keep routing information to all nodes in the network, not subject to any existing data flow. OLSR is a link state protocol [58] using an optimized broadcast mechanism for the dissemination of link state information. In *reactive routing* (e.g., AODV [45]), routes are found on demand and nodes find routes to their destinations as they are needed. Route discovery starts by broadcasting a *route request* (RREQ) message throughout the network. This message is relayed until it reaches a node with a valid route to the destination, or the destination itself. Once this happens, a *route reply* (RREP) message is sent back to the source by reversing the path traversed by the RREQ message. Only after receiving the corresponding RREP message can the source start sending packets to the destination.

Reactive and proactive routing can be combined, resulting in *hybrid protocols* (e.g., ZRP [23]). In this case, routes to some nodes (usually the nearest ones) are kept proactively, while routes to the remaining nodes are found on-demand.

## 1.3 Domination in Graph Theory

Since our contributions involve the computation of dominating sets, we provide a brief introduction to domination in graph theory below.

In our notation, the undirected graph $G = (V, E)$ consists of a set of vertices $V = \{n_1, \ldots n_k\}$, and a set of edges $E$ (an edge is a set $\{n_i, n_j\}$, where $n_i, n_j \in V$ and $n_i \neq n_j$). A set $D \subseteq V$ of vertices in a graph $G$ is called a *dominating set* (DS) if every vertex $n_i \in V$ is either an element of $D$ or is adjacent to an element of $D$ [24]. If the graph induced by the nodes in $D$ is connected, we have a *connected dominating set* (CDS). The problem of computing the minimum cardinality DS or CDS of any arbitrary graph is known to be NP-complete [19].

A variety of conditions may be imposed on the dominating set $D$ in a graph $G = (V, E)$. Among them, there are *multiple domination,* and *distance domination* [24]. *Multiple domination* requires that each vertex in $V - D$ be dominated by at least $k$ vertices in $D$ for a fixed positive integer $k$. The minimum cardinality of the dominating set $D$ is called the *k-domination number* and is denoted by $\gamma_k(G)$. *Distance domination* requires that each vertex in $V - D$ be within distance $r$ of at least one vertex in $D$ for a fixed positive integer $r$. In this case, the minimum cardinality of the dominating set $D$ is called the *distance-r domination number,* and is denoted by $\gamma_{\leq r}(G)$.

Henning et al. [26] have presented some bounds on the *distance-r domination number* $\gamma_{\leq r}(G)$. They show that, for an integer $r \geq 1$, if graph $G$ is a connected graph of order $n \geq r + 1$, then $\gamma_{\leq r}(G) \leq \frac{n}{r+1}$. An algorithm that computes a *distance-r dominating set* within the established bounds is also presented.

The $(k, r)$-DS problem is defined [31] as the problem of selecting a minimum car-

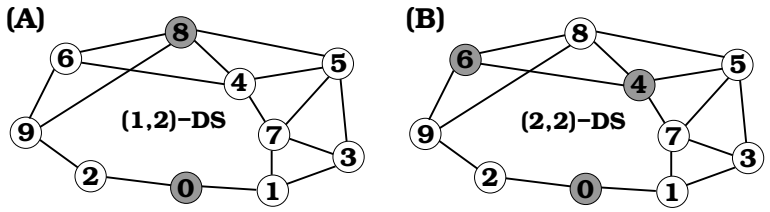



**Figure 1.2**: *Dominating set* examples (gray nodes are *dominating*): (A) $(1, 2)$-DS, dominated nodes are *at most* two hops from *at least* one dominating node. (B) $(2, 2)$-DS, dominated nodes are *at most* two hops from *at least* two dominating nodes.

dinality vertex set $D$ of a graph $G = (V, E)$, such that every vertex $u$ not in $D$ is at a distance smaller than or equal to $r$ from at least $k$ vertices in $D$. The problem of computing a $(k, r)$-DS of minimum cardinality for arbitrary graphs is also NP-complete [31]. Figure 1.2 shows some $(k, r)$-DS examples.

Joshi et al. [31] have provided centralized solutions for solving the $(k, r)$-DS problem in *interval graphs* (IG). A graph $G$ is said to be an interval graph if there is a one-to-one correspondence between a finite set of closed intervals of the real line and the vertex set $V$, and two vertices $u$ and $v$ are said to be connected if and only if their corresponding intervals have a nonempty intersection. Even though the solutions presented by Joshi et al. [31] are optimal, IGs are limited to very simple network topologies.

## 1.4 Research Contributions and Summary of Results

Chapter 2 summarizes prior work related to the research results reported in this thesis. Chapter 3 proposes a novel algorithm for improving broadcast operations in MANETs. The algorithm computes a dynamic source-based dominating set of the network. The solution is shown to reduce the number of broadcast transmissions necessary to flood the network. To

show its applicability to MANETs, we implemented the algorithm in a simulator as part of the route discovery process in an on-demand routing protocol. Redundant broadcasts increase the number of packet collisions, and consequently delay the response for RREQs in the route discovery process. Because the new protocol significantly reduces the total number of RREQ transmissions, this translates into an increased delivery ratio, smaller end-to-end delays for data packets, lower control overhead, and fewer collisions of packets.

Our analysis of the route discovery process in on-demand routing protocols led us to realize that it could benefit from a particular configuration of *connected dominating sets*. Most heuristics for distributively computing dominating sets require that nodes learn only their two-hop neighborhood. When a broadcast protocol based on neighbor information is used, it is possible to maintain fresh routes to all nodes within two hops, because every node has the two-hop neighborhood information. In this case, it is not necessary to broadcast the *route request* (RREQ) packet to every node in the network: disseminating it to a connected dominating set with the property that nodes are at most two-hops from a dominating node (i.e., a *two-hop connected dominating set* (TCDS)) is enough.

Chapter 4 presents *three-hop horizon pruning* (THP), which is the first distributed solution for computing a TCDS of the network. THP is shown to outperform the best existing heuristics presented in the literature, when a TCDS is preferred over a CDS. We also integrated the algorithm into an on-demand routing protocol for enhancing the route discovery process. The results of simulation studies indicated that the new approach improves the performance in all aspects for low mobility scenarios. To improve protocol performance in high mobility scenarios, we introduce enhancements to the basic design, and show that with these

enhancements the protocol outperforms other protocols in all mobility scenarios considered in the simulations.

The work done on THP motivated the design of a framework using dominating sets for building flexible hierarchies. Chapter 5 presents our solution, which builds structures that cover a node in the network with a minimum number of dominating nodes and a maximum distance to the dominating nodes. In terms of *domination in graphs*, this approach integrates *multiple domination* (minimum number of dominating nodes) and *distance domination* (maximum distance to dominating nodes). For example, one could build a structure such that every node in the network is covered by at least $k$ leaders at most $d$ hops distant. A structure like this could be used to support operations with increased redundancy (by increasing the number of leaders), and an adjustable degree of availability (by setting the maximum distance to the leaders appropriately). We present the first centralized and distributed solutions to this problem. The centralized solution, KR, provides an approximation to the optimum solution, which is known to be NP-Complete, and serves as a lower bound when evaluating the distributed solution. The distributed solution, DKR, is applicable to MANETs, because it relies on partial topology information.

Chapter 6 shows an application for the above framework, which consists of a novel multi-core multicast protocol for MANETs using DKR for core-election. The dominating set computed via the distributed algorithm is used for assigning cores. Nodes are equipped with two interfaces, one for general communication, and the other for communication between cores and their members. Cores transmit packets to their members on a specific non-interfering channel via the dedicated interface, and receivers listen in the corresponding chan-

nel in the same interface. To reach all members with a single transmission, cores transmit packets with a larger power, such that all member within $r$-hops from the cores can successfully receive the packet. Therefore, all packets transmitted by the cores are expected to be successfully received by the members. The distributed algorithm is shown to perform well for electing cores, and the new multicast protocol is shown to outperform one of the best known multicast protocols presented in the literature, the *protocol for unified multicasting through announcements* (PUMA) [63].

The algorithms presented in this thesis for computing a CDS of the network can be applied to any other MANET protocol requiring flooding of control messages. THP can also be applied to any on-demand routing protocol which is based on the dissemination of RREQ messages. Likewise, DKR functions as a framework for building a large variety of hierarchical structures, and can be tailored to the target application (e.g, structures having multiple access-points to the outside network, and sensor networks with multiple sinks per sensor). Thus, while we present this work in the context of improving specific protocols, it is far more general than these case-by-case improvements. The work presented here is a general purpose toolkit for improving a wide range of coordination protocols in MANETs along multiple axes of performance. While we mostly focus on improvements to AODV, other protocols could benefit from this approach, such as *dynamic source routing* (DSR) [30], and *on demand multicast routing protocol* (ODMRP) [33].

Several papers [50–55] based on the above research work have been published or accepted for publication. Chapter 7 summarizes our contribution.

# Chapter 2

# Related Work

The focus of this thesis is on the design and analysis of algorithms to improve the coordination of nodes in MANETs. Before proceeding to our contributions, this Chapter presents the work related to our research.

Given that broadcasting of signals is the underlying mechanism for coordination, Section 2.1 presents the most significant research results on broadcasting in MANETs. Because our work can be applied to most routing protocols in MANETs, Section 2.2 presents a brief review of the most cited routing protocols in the literature. Organization of nodes is another form of coordination, including *clustering* and *topology control* as the two representatives. Section 2.3 presents the most significant results regarding these topics. In Section 2.4 we present the work related to core-based multicast routing, because we apply our general clustering approach to a novel multi-core multicast protocol.

## 2.1 Broadcasting in MANETs

Several broadcasting techniques have been proposed, differing among each other on the heuristics applied to reduce the redundancy on broadcast transmissions. Broadcasting protocols can be categorized into the following four classes [67]:

*Blind flooding* [41]: Each node broadcasts a packet to its neighbors whenever it receives the first copy of a broadcast packet; therefore, all nodes in the network broadcast the packet exactly once.

*Probability-based methods* [61]: A node re-broadcasts a packet with a given probability $p$ (if $p = 1$, we have blind flooding).

*Area-based methods* [61]: A node broadcasts a packet based on the information about its location and the location of its neighbors (e.g., if a node receives the packet from a neighbor really close to it, probably it will not reach other nodes other than the nodes reached by the first broadcast).

*Neighbor information methods* [36]: In these methods, a node has partial topology information, which typically consists of the topology within two hops from the node (two-hop neighborhood). There are two main classes of methods in this category. In a *neighbor-designated method* a node that transmits a packet to be flooded specifies which one-hop neighbors should forward the packet. In a *self-pruning method* a node simply broadcasts its packet, and each neighbor that receives the packet decides whether or not to forward the packet.

Williams and Camp [67] have shown that *neighbor information* methods are preferred over other types of broadcast protocols. Between the two classes of neighbor informa-

tion methods, Lim and Kim [36] show that the simplest form of neighbor-designated algorithm outperforms the simplest form of self-pruning, and Wu and Dai [68] show that an improved self-pruning technique outperforms the most efficient neighbor-designated algorithm (both algorithms based on the two-hop neighborhood information).

Dominating sets play a major role in deciding the forwarding list in neighbor-designated algorithms. Extensive work has been done on finding good approximations for computing the *minimum cardinality* CDS (MCDS). An algorithm with a constant approximation of eight has been proposed by Wan et al. [64]. However, their approach requires that a spanning tree be constructed first in order to select the dominating nodes (forwarding nodes), and only after the tree has been constructed a broadcast can be performed.

Lim and Kim [36] show that the MCDS problem can be reduced to the problem of building a *minimum cost flooding tree* (MCFT). Given that an optimal solution for the MCFT problem is not feasible, they propose heuristics for flooding trees, resulting in two algorithms: *self-pruning* and *dominant pruning* (DP). They show that both algorithms perform better than *blind flooding*, and that DP outperforms the simplest form of *self-pruning*.

DP [36] is a neighbor-designated method (i.e., the sending node decides which adjacent nodes should relay the packet). The forwarding nodes are selected using the *greedy set cover* (GSC) algorithm. GSC recursively chooses one-hop neighbors that cover the most two-hop neighbors, repeating the process until all two-hop neighbors are covered. The identifiers (IDs) of the selected nodes are piggy-backed in the packet as the forwarding list. A receiving node that is requested to forward the packet again determines the forwarding list.

*Multi-Point Relay* (MPR) [14] is another efficient broadcast technique that is simi-

lar to DP. MPR also applies GSC in the selection of dominating nodes. However, MPR first chooses as forwarders those candidates that have exclusive coverage of some two-hop neighbor, and only then apply GSC over the remaining nodes. MPR is used for reducing duplicate transmissions of control packets (i.e., link state information) in the *Optimized Link State Routing* (OLSR) protocol [14].

A few enhancements to dominant pruning have been reported recently [37]. Lou and Wu [37] propose two enhancements to DP: *total dominant pruning* (TDP), and *partial dominant pruning* (PDP). TDP requires that the two-hop neighborhood of the sender be piggybacked in the header of the packet. This information reduces the size of the two-hop neighbor set that needs to be covered by the forwarders. The header size increases proportionally to the number of nodes in the two-hop neighborhood, which may become a problem in dense networks. PDP enhances DP by eliminating the two-hop nodes advertised by a neighbor shared by both the sender and the receiver (forwarder). Simulation results assuming an ideal MAC layer in which no contention or collisions occur show that both TDP and PDP improve DP in a static environment. A dynamic scenario is also evaluated, and DP is shown to perform better than both TDP and PDP.

A general framework for self-pruning has been reported by Wu and Dai [68], who proposed two approaches for broadcasting through self-pruning, one static and another dynamic. In the static approach, a CDS is constructed based on the network topology, but not relative to any broadcasting. In the dynamic approach, a CDS is constructed for a particular broadcast, and its result depends on the source and the progress of the broadcast process. For both approaches, two *coverage conditions* are presented: Coverage Condition I (CC-I), and

Coverage Condition II (CC-II).

In CC-I, a node $n_i$ does not broadcast the packet if for any two neighbors $n_j$ and $n_k$, there is a path connecting them via several intermediate nodes with either higher priority values (e.g., node degree, node IDs) than node $n_i$, or with visited node status (i.e., the $h$ most recently visited nodes are included in the packet header). In CC-II, a node $n_i$ does not broadcast the packet if the node has a *coverage set*, which is defined as a set of neighbors with either higher priority values than node $n_i$, or with visited node status, such that all two-hop neighbors of node $n_i$ are covered by the *coverage set*.

Wu and Dai showed that CC-I performs better than CC-II when node IDs are used as priority values, and when node degrees are used as priority values they present similar results. They also showed that there is a tradeoff between efficiency and overhead, and that CC-I with two-hop neighborhood information, two-hop routing history, and node degrees as priority values (referenced as the *Base* configuration), outperforms the best neighbor-designated algorithm (i.e., TDP).

Several other existing algorithms (i.e., Rules 1 and 2 [72], Stojmenovic's algorithm [56], Rule *k* [15], Span [10], and LENWB [57]) were shown [68] to be special cases in the general framework. Simulation results show that the Base configuration outperforms all the others, but the difference amongst Base, Span, and LENWB is marginal. The neighborhood size is also analyzed, and it is shown that a neighborhood size larger than three hops does not add much power to the coverage conditions. In other words, the coverage conditions do not reduce the average number of forwarding nodes for an increasing size of the neighborhood information.

Wu and Dai [69] further analyzed the coverage conditions they reported previously [68] and showed that several other algorithms can be derived from the generic framework. The impact of four implementation issues, namely timing (static or dynamic), selection (self-pruning, neighbor-designated, and hybrid), space (network topology information), and priority (e.g., node ID, node degree), is analyzed. It is also shown that self-pruning and neighbor-designated algorithms can be combined together forming hybrid algorithms.

All distributed algorithms that rely on knowledge of the two-hop neighborhood are prone to error in the presence of mobility. The main reason is that nodes may have inconsistent information about the neighborhood, compromising network connectivity. Wu and Dai [71] propose a solution to address the link availability problem using two transmission ranges. Information about the neighborhood and the set of forwarders is computed using a smaller radio range. The broadcast process is performed using a larger radio range. The objective is to give nodes a *buffer zone* in which they can move without compromising local connectivity.

## 2.2 Routing in MANETs

In MANETs, routes can be computed proactively (i.e., nodes keep routes to all nodes in the network, regardless of any data flow), or reactively (i.e., routes are computed as needed). However, both approaches make use of flooding of control messages; either to search for a route on demand, or to keep accurate routing information at all times. In this Section, we present the most representative routing protocols in each of these two categories.

### 2.2.1 Proactive Routing

Ogier et al. [42] proposed the *topology broadcast reverse path forwarding* (TBRPF) protocol. In TBRPF, each node only advertises the part of its network graph used for reachability, which minimizes the number and size of advertisements. Based on the partial graph, each node runs a modified Dijkstra algorithm to compute a source tree. Each node also has the option to report additional topology information (up to the full topology). TBRPF applies an optimized flooding mechanism to reduce the overhead incurred by topology information dissemination.

*Source-tree routing* [18] (STAR) is a link-state protocol that achieves low overhead by not reporting all link states. Each node constructs a source tree – with the current node as the source – to each known destination. The node then reports the source tree to neighbors, which iteratively build source trees and report the trees. This allows each node to build a connected graph using a subset of edges. STAR has two modes of operation. In the *optimal routing approach* (ORA), STAR behaves like conventional link state protocols and notifies neighbors on any change to the source-tree. This allows nodes to select near optimal routes. In the *least-overhead routing approach* (LORA), nodes only send updates when a new destination appears, a known destinations become disconnected, or the source-tree changes in a way that could potentially cause a loop.

The OLSR protocol [14] introduces several techniques to reduce the overhead associated with flooding link-state advertisements over a wireless ad hoc network. OLSR uses three optimizations. The key concept of OLSR is the application of MPR for flooding control traffic over the network. Each node, $X$, selects MPRs from its 1-hop neighbor set. Node

$X$ is called a *selector* of nodes in the MPR set because node $X$ has selected those nodes as its MPRs. The first optimization is that only MPRs forward broadcast control traffic from its selectors. This reduces the number of redundant broadcast transmissions for flooding the network. The second optimization is that only MPRs generate link-state advertisements. This reduces the number of control packets. The third optimization is that an MPR only needs to generate link-state advertisements for its selectors, not for all 1-hop neighbors.

### 2.2.2 Reactive Routing

DSR [30] builds complete hop-by-hop routes at each source node. DSR works by broadcasting a RREQ message over the network and recording the path of the packet. When a node with a path to the destination receives the request, it can send a route reply along the reverse route. The reply contains the responding node's path and records its route back to the requesting node. Thus, the requesting node has the complete path. As nodes process RREQ messages and RREP messages, nodes learn paths to other nodes, and store this information in a local cache. The main disadvantage of this approach is that route entries may become stale quickly depending on the mobility of nodes. As a consequence, a node may waste time trying to route through stale paths until the node decide to discard the stale entries.

Like DSR, AODV [45] also makes use of RREQ and RREP messages. The main difference is that AODV does not use source routing. In order to guarantee loop-freedom, destination sequence number are used to eliminate stale information. When a node transmits an advertisement, it includes an increasing sequence number for itself. Each routing entry stored at a node also includes the sequence number the destination attached to the advertisement.

This allows nodes to reject stale advertisements. A key element in AODV is that when a node detects that a route becomes invalid, the node must increment the stored destination sequence number. It also makes the node immune to receiving stale information already in the network that the node had previously issued.

## 2.3 Clustering and Topology Control

*Clustering* is the problem of building a hierarchy among nodes [12]. The substructures that are collapsed in higher levels are called *clusters*. Given a graph $G = (V, E)$, the clustering process initially divides $V$ into a collection of subsets $\{V_1, V_2, \ldots, V_i\}$, where $V = \bigcup_{l=1}^{i} V_l$, such that each subset $V_l$ induces a connected subgraph of $G$, or a *cluster*. Subsets do not need to be disjoint (i.e., subsets can overlap). In each cluster, at least one node may represent the cluster, and this node is usually called a *cluster-head*. The network can then be abstracted such that any cluster-head connects to another cluster-head whenever there is at least one node in each cluster directly connected to each other.

There are three main approaches for topology control in wireless ad hoc networks. The first is topology control based on *geometry structures* used to define a topology for a network, such as the *relative neighborhood graph* (RNG) [60], *gabriel graph* (GG) [16], or $\theta$-graph [38]. Topology control based on *transmission power control* [28, 40] adjusts the transmission power on a per node basis. Hierarchical topology control [8] selects a subset of nodes to form the backbone of the network, and the backbone is usually computed using some distributed CDS algorithm.

Topology control based on hierarchies [8,34] and clustering are closely related problems. While the former defines a real *backbone* of the network (i.e., the backbone forms a CDS of the network), the latter constructs a *virtual backbone* (i.e., the set of cluster-heads forms a DS of the network).

The basic requirement for any topology control mechanism is to guarantee that the topology is connected. That is, any pair of nodes that are connected in the original network must still be connected in the topology built on top of the network. For clustering, nodes in a cluster are connected through the cluster-head; and border nodes (i.e., nodes that belong to two or more clusters, or that have at least one neighbor in a different cluster than its own) connect clusters to each other.

Clustering based on domination in graphs has been explored extensively. Baker and Ephremides [4] devised one of the first clustering algorithms based on domination. Chen et al. [11] explore *independent dominating sets* (IDS) for computing cluster such that cluster-heads are at least $k + 1$ hops from each other. However, the use of IDS for clustering is not recommended when the topology changes, because cluster-head changes may propagate throughout the network, an effect that has been called the *chain reaction* [20].

*Max-Min* [3] is an election-based distributed solution for the $d$-hop DS problem (i.e., distance $d$ domination), which takes $2r$ rounds to complete. Cluster-heads are computed during the first $r$ rounds, and nodes affiliate to their dominating nodes during the subsequent $r$ rounds. The authors also show that the problem of computing the minimum $r-$hop dominating set is NP-complete for unit-disk graphs [13].

Liang and Hass [35] proposed a distributed algorithm to compute $d$-hop DS. The

algorithm is a distributed version of *Greedy Set Cover* (GSC), producing dominating sets with the same cardinality as the centralized solution for this problem. However, their solution requires the $2r$-hop neighborhood information. In MANETs, keeping information about the neighborhood becomes more difficult as we increase the distance to the neighbors.

## 2.4 Core-Based Multicast Routing

Multicast routing protocols can be classified as tree-based and mesh-based. Tree-based can be further classified as single-source, shortest-path trees and shared, core-based trees. Core-based trees are more scalable compared to shortest-path trees, but usually present higher end-to-end delay and poor fault tolerance.

To improve the performance of core-based trees, multiple cores are deployed. Because more than one core is simultaneously active, the protocol tolerates core failure. The distribution of cores in the network has a direct impact on the performance, since cores placed closer to the receivers can reduce the end-to-end delay.

With multiple cores there are two *one-to-all* designs [74]: *senders-to-all*, and *members-to-all*. In *senders-to-all*, senders transmit to all cores, and members join to just one core (usually the nearest one). In *members-to-all*, senders select one of the cores to send their data packets, and members need to join all cores. As for an *one-to-one* approach, each of the cores must join to at least one other [5]. In this case, senders send to just one of the cores, and receivers join to just one of the cores.

Senders-to-all has several advantages compared to members-to-all. Both approaches

use one tree per core, but in members-to-all each tree connects all members, increasing the routing state in each router. In senders-to-all members decide which core to join, allowing members to choose the core that better satisfy their requirements (e.g., lower end-to-end delay).

While the *one-to-one* approach combines advantages of the two *one-to-all* designs, it requires a reachability and maintenance protocol between the cores. Failure between any two virtually adjacent cores partitions the network.

Core placement has a direct impact on the performance of the protocol. If the number of cores is fixed, say $k$ cores, then the problem is referred as $k$-center [19], and is defined as the problem of locating $k$ cores in the network such that the distance from nodes to the cores is minimized. If the number of cores is not fixed, but the maximum distance to a core, say $r$, is fixed, then the problem is the same as the problem of computing $d$-hop *dominating sets* (DS) in graphs. Both problems are known to be NP-Complete.

A natural greedy solution to the $(1, r)$-DS problem has been applied for *core* placement in multicast trees with multiple cores [74]. This solution is intended for wired networks, and requires knowledge of the entire topology. Performance results show that this approach reduce the end-to-end delay of multicast data packets.

# Chapter 3

# Enhanced Dominant Pruning (EDP)

On-demand route discovery is based on *route request* (RREQ) and *route reply* (RREP) messages (e.g., AODV [45] and DSR [30]). The way in which these messages are handled may differ among different protocols, but their functionality remains the same: a request is relayed until it reaches a node with a valid route to the destination or the destination itself, which triggers a reply message sent back to the originator. Several parameters (such as how long to keep requests in a cache, timeouts for requests, timeouts for hellos, and the like) are subject to tuning, and the choices made may result in improvements in the protocol performance. However, RREQs are propagated using either an unrestricted broadcast or an expanding ring search [48]. In either case, the resulting flooding operation causes considerable collisions of packets in wireless networks using contention-based channel access.

We present an approach to reduce the number of broadcast messages at the expense of having to attach more information in the header of the control packets. We call our proposal *Enhanced Dominant Pruning* [50] (EDP), which can be applied to any on-demand routing

protocol that relies on broadcasting control packets when searching for a route to a given destination. To show the applicability of EDP to an existing protocol, we have implemented EDP in AODV. Nodes use hello messages to disseminate their valid one-hop neighbors for building the two-hop neighborhood, which is the minimum requirement for the connected dominating set algorithm under consideration.

## 3.1 Dominant Pruning Review

We use a simple graph, $G = (V, E)$, to represent an ad hoc wireless network, where $V$ represents a set of wireless mobile hosts (nodes) and $E$ represents a set of edges (links). The network is seen as a *unit disk graph* [13], i.e., the nodes within the circle around node $v$ (corresponding to its radio range) are considered its neighbors.

In *dominant pruning* (DP) [36] the sending node decides which adjacent nodes should relay the packet. The relaying nodes are selected using a distributed CDS algorithm, and the identifiers (IDs) of the selected nodes are piggybacked in the packet as the forwarder list. A receiving node that is requested to forward the packet again determines the forwarder list. The flooding ends when there is no more relaying nodes.

Nodes keep information about their two-hop neighborhood, which can be obtained by the nodes exchanging their adjacent node list with their neighbors. DP is a distributed algorithm that determines a set cover based on the partial knowledge of the two-hop neighborhood. Ideally, the number of forwarding nodes should be minimized to decrease the number of transmissions. However, the optimal solution is NP-complete and requires that nodes know the entire topology of the network. DP uses the *greedy set cover* (GSC) algorithm to determine

the forwarder list of a packet (i.e., the list of nodes that should forward the packet) based just on partial knowledge of the network topology. GSC recursively chooses one-hop neighbors that cover the most two-hop neighbors, repeating the process until all two-hop neighbors are covered.

The set of nodes within two-hops from node $n_i$ is denoted by $N_2^i$, and the set of one-hop neighbors of node $n_i$ is denoted by $N_1^i$. If node $n_i$ is the source of the broadcast, it determines its forwarder list so that all nodes in $U_i = N_2^i - N_1^i$ receive the packet. The set of forwarder nodes is denoted by $F_{DP}^i = \{f_1, f_2, ..., f_m\} \subseteq N_1^i$, such that $\bigcup_{f_k \in F_{DP}^i} (N_1^{f_k} \bigcap U_i) = U_i$. A forwarder node $n_j \in F_{DP}^i$ determines its own forwarder list upon receiving the broadcast. Node $n_j$ does not need to cover the neighbors of node $n_i$ (i.e., $N_1^i$), because they were already covered by the previous broadcast. In this case, $U_j = N_2^j - N_1^j - N_1^i$ is the set to be covered. The set $F_{DP'}^j \subset N_1^j$ is the temporary set cover of node $n_j$. Our solution includes the set of one-hop neighbors shared by nodes $n_i$ and $n_j$ (i.e., $N_1^j \bigcap N_1^i$), in the first part of the computation of the forwarder list. The final forwarder list is defined as $F_{DP}^j = F_{DP'}^j - F_{DP}^i$.

The solution presented by Lim and Kim [36] is incorrect, because only nodes in the subset $N_1^j - N_1^i$ are considered for the computation of the forwarder list, which can lead to incorrect results for particular topologies. The reason is simple, nodes being shared by the source and the receiver are still candidates as forwarder nodes, because node $n_j$ may have two-hop nodes exclusively advertised by some shared node. Because a node knows the sender's forwarder list, it can get rid of those nodes that were previously chosen as forwarder nodes by the sender. It turns out that the resulting forwarder list described in [36] is in fact in the subset $N_1^j - N_1^i$.

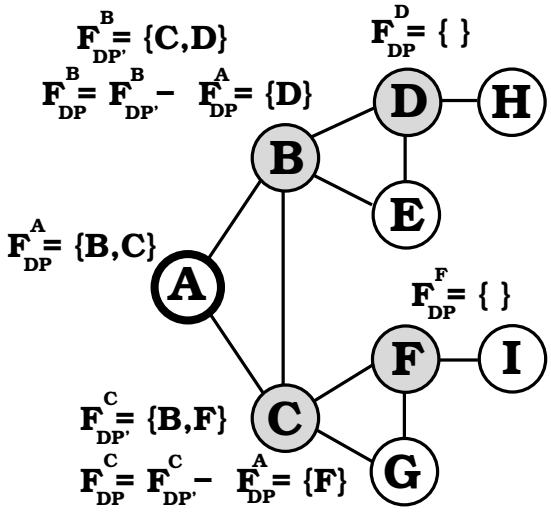



**Figure 3.1**: Example where original DP fails (node A is the source).

Figure 3.1 shows an example where DP as proposed in [36] fails. Consider node $A$ as the source of the broadcast. Node $A$ selects nodes $B$ and $C$ as the forwarder nodes. Note that $U_B = \{F, G, H\}$, but there is no node in the set $\{D, E\}$ (i.e., $N_1^B - N_1^A$) that can cover the set $\{F, G\}$. The same can be said for $U_C = \{D, E, J\}$, where there is no node in the set $\{F, G\}$ (i.e., $N_1^C - N_1^A$) that can cover the set $\{D, E\}$.

Our solution guarantees that nodes in $N_1^A \bigcap N_1^B$ take part in the selection of the forwarder nodes of $B$, and that nodes in $N_1^A \bigcap N_1^C$ take part in the selection of the forwarder nodes of $C$. It is just a matter of consistency, even though some nodes in $F_{DP'}^B$ and in $F_{DP'}^C$ are ruled out of the resulting forwarder list when they were already in the sender's forwarder list.

## 3.2 Enhanced Dominant Pruning

The objective of *Enhanced Dominant Pruning* (EDP) is to reduce the number of broadcast packets necessary to flood the network with the same guarantees provided by DP (after applying the modifications cited previously). In the following, we assume that a neighbor protocol is available to provide the two-hop neighborhood information.

The *EDP forwarder list* as determined by node $n_i$ is denoted by $\mathcal{F}_i$. We use the term *EDP forwarder list* to emphasize that the resulting list might be different from the one obtained by simply running DP (i.e., $F^i_{DP}$) over $U_i$. The current node is denoted by $n_i$, and the node that sent the packet is denoted by $\mathsf{S}$ (if the current node is the source of the broadcast, then $\mathsf{S} = \emptyset$). The sender's forwarder list is denoted by $\mathcal{F}_\mathsf{S}$, and the second-to-previous forwarder list is denoted by $\mathcal{F}_{\mathsf{S}_\mathsf{S}}$. $\mathcal{F}_\mathsf{S} = \emptyset$ if $n_i$ is the source of the broadcast. In a similar manner, $\mathcal{F}_{\mathsf{S}_\mathsf{S}} = \emptyset$ if $n_i$ is the source of the broadcast or if the sender $\mathsf{S}$ is the source. The packet header must specify the forwarder list and the sender's forwarder list (the sender $\mathsf{S}$ of the packet is obtained from the packet header), but that should not be a problem given that both lists are expected to be small, because GSC is applied to the two-hop neighborhood.

Figure 3.2 shows the pseudo-code for determining the *EDP forwarder list* $\mathcal{F}_i$. Let $\mathcal{C}$ be the set of neighbors of node $n_i$ that are also in the sender's forwarder list $\mathcal{F}_\mathsf{S}$. Let $\bigcup_{n_k \in \mathcal{C}} N_1^k$ be the set of nodes adjacent to neighbors that are also in the sender's forwarder list. These nodes do not need to be considered when running DP, because they are guaranteed to be covered by some other forwarder node. Let $\mathcal{I}$ be the set of nodes in $\mathcal{F}_\mathsf{S}$ with identifiers larger than node $n_i$. The set of forwarder nodes of $n_i$ that are reachable through other forwarder node in the sender's list (i.e., there is a disjoint two-hop path to node $n_k$ through another node

in $\mathcal{F}_{\mathsf{S}}$ with a higher priority) is denoted by $\mathcal{P}$. The set of neighbors that are already covered by nodes in $\mathcal{F}_{\mathsf{S_s}}$ (the second-to-previous forwarder node) is denoted by $\mathcal{Q}$.

**Figure 3.2**: Enhanced Dominant Pruning

**Data**: $n_i$, $\mathcal{F}_{\mathsf{S}}$, $\mathcal{F}_{\mathsf{S_s}}$, $U_i$
**Result**: $\mathcal{F}_i$, the forwarder list
**begin**

1 $C \longleftarrow N_1^i \bigcap \mathcal{F}_{\mathsf{S}}$
2 $U_{DP}^i \longleftarrow U_i$
3 **for** $n_k \in \mathcal{C}$ **do**
4 $U_{DP}^i \longleftarrow U_{DP}^i - N_1^k$
5 $F_{DP}^i \longleftarrow DP(U_{DP}^i)$
6 $\mathcal{I} \longleftarrow \emptyset$
7 **for** $n_k \in \mathcal{F}_{\mathsf{S}}$ **do**
8 **if** $n_k > n_i$ **then**
9 $\mathcal{I} \longleftarrow \mathcal{I} \bigcup \{n_k\}$
10 $\mathcal{P} \longleftarrow \emptyset$
11 **for** $n_k \in F_{DP}^i$ **do**
12 **for** $n_l \in \mathcal{I}$ **do**
13 **if** $n_l \in N_1^k$ **then**
14 $\mathcal{P} \longleftarrow \mathcal{P} \bigcup \{n_k\}$
15 $\mathcal{Q} \longleftarrow \emptyset$
16 **for** $n_k \in N_1^i$ **do**
17 **for** $n_l \in N_1^k$ **do**
18 **if** $n_l \in \mathcal{F}_{\mathsf{S_s}}$ **then**
19 $\mathcal{Q} \longleftarrow \mathcal{Q} \bigcup \{n_k\}$
20 $\mathcal{F}_i \longleftarrow F_{DP}^i - \mathcal{Q} - \mathcal{P} - \mathcal{F}_{\mathsf{S}} - \mathcal{F}_{\mathsf{S_s}}$

**end**

As in DP, a forwarding node does not need to include in its forwarder list those neighbors that are also neighbors of the sender $\mathsf{S}$ (i.e., $N_1^i \bigcap N_1^{\mathsf{S}}$). Because the sender $\mathsf{S}$ already sent the packet to all its neighbors, all the common neighbors between the sender and the receiver can be excluded from the forwarder list. Neighbors that are also in the sender's forwarder list $\mathcal{F}_{\mathsf{S}}$ (line 1) can have their one-hop nodes removed from $U_i$ (lines 2 through 4).

Then DP is run on this reduced set, denoted by $U^i_{DP}$ (line 5).

A node in $F^i_{DP}$ that is covered by at least one more node in $\mathcal{F}_\mathsf{S}$ needs to be covered by just one of these nodes. To select one, we use node identifiers as priorities, and the node with the largest ID wins. Lines 7 through 9 present the pseudo-code that creates the set $\mathcal{I}$, which contains the nodes in $\mathcal{F}_\mathsf{S}$ with identifiers larger than the local node $n_i$. The set $\mathcal{P} \subset F^i_{DP}$ has the forwarder nodes that are reachable through another node in the sender's forwarder list (lines 10 through 14). That is, there is a disjoint two-hop path to node $n_k \in F^i_{DP}$ through another node in $\mathcal{F}_\mathsf{S}$ with a higher priority. Therefore, a node in the set $\mathcal{P}$ can be excluded from the forwarder list.

A neighbor $n_k$ that was previously chosen as a forwarder node by the second to previous node (i.e., $n_k \in \mathcal{F}_{\mathsf{S}_\mathsf{S}}$), and neighbors covered by a node in $\mathcal{F}_{\mathsf{S}_\mathsf{S}}$, can be removed from the forwarder list (lines 15 through 19). A neighbor $n_k$ is covered by some node $n_l \in \mathcal{F}_{\mathsf{S}_\mathsf{S}}$ if $n_l \in N^k_1$. Finally, the EDP forwarder list $\mathcal{F}_i$ is updated on line 20.

Consider the example shown in Figure 3.3. Node $A$ selects nodes $\{B, D, F\}$ for its forwarder list. Node $D$ selects nodes $\{E, G\}$ as forwarders, because $E$ is the only neighbor covering node $C$, and node $G$, because it is the only neighbor covering nodes $\{H, J\}$. Node $G$ can be removed from $D$'s forwarder list, because node $G$ is covered by another forwarder node with a higher priority (i.e., node $F \in \mathcal{F}_A$, and $ID(F) > ID(D)$). On the other hand, node $F$ selects node $G$ as its forwarder node, because node $F$ wins over node $D$. Node $G$ selects node $D$ as its forwarder, because $D$ is the only neighbor covering node $E$, but node $D$ can be dismissed because node $D$ is in the second to previous forwarder list (i.e., $D \in \mathcal{F}_{\mathsf{S}_\mathsf{S}} = \mathcal{F}_A$). Node $B$ selects node $C$ as its forwarder, because node $C$ is the only neighbor to cover node

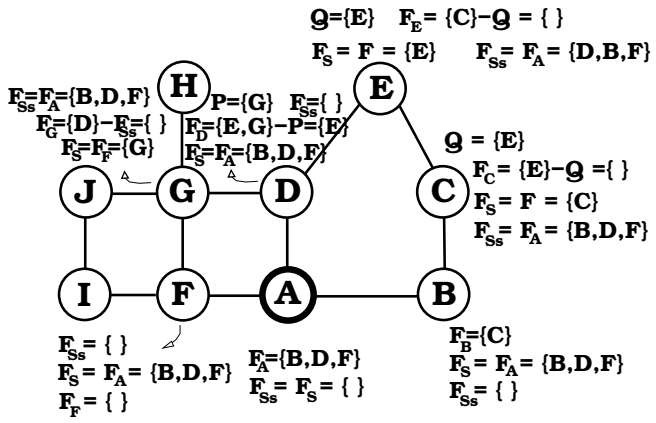



**Figure 3.3**: Node A is the source. The knowledge about the second to previous forwarder list ($\mathcal{F}_A$) allows nodes E and C to exclude each other from their forwarder list, and node G to exclude node D. Node D reduces the size of its forwarder list by using the information provided by the set $\mathcal{P}$.

$E$. Node $C$ determines $E$ as a forwarder node, because $E$ is the only neighbor covering node $D$. Node $C$ does not need to include node $E$ in the forwarder list, because node $E$ is covered by a node in $\mathcal{F}_A$. The same happens at node $E$, which selects node $C$ as a forwarder node but it is not necessary to include node $C$ in the forwarder list, because node $C$ is covered by node $B$. It is important to note that, in order to exclude a neighbor from the forwarder list, it suffices that the node is covered by other node in $\mathcal{F}_{S_S}$. It is not a requirement that the excluded node be chosen as a forwarder node by the node covering it.

Consider the example illustrated in Figure 3.4. Node $A$ is the source of the broadcast. Nodes $B$ and $C$ are chosen as forwarders. Node $B$ does not need to cover nodes $F$ and $G$ because they are adjacent to other node (i.e., node $C$) in the sender's forwarder list. Given that, node $B$ determines node $D$ as its forwarder node, which in turn determines node $E$ as its forwarder node. Nevertheless, $D$ does not need to forward the packet to node $E$ because it is covered by a previous forwarder node (i.e., node $B$ chosen as forwarder node by node $A$), and node $E$ is adjacent to both node $D$ and the sender $B$. In a similar manner, node $C$ does

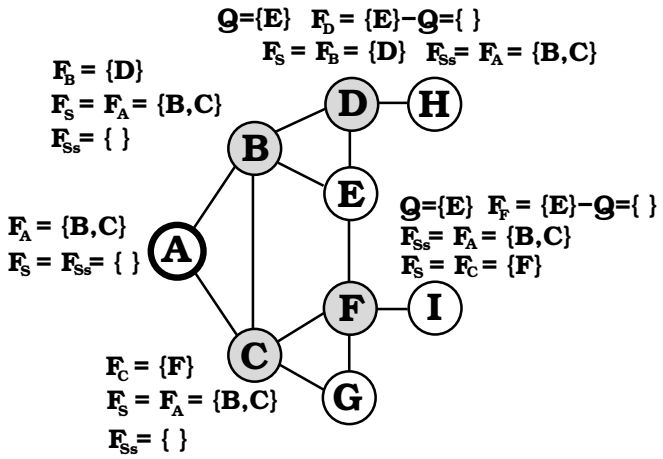



**Figure 3.4**: Node A is the source. Nodes B,C,D,F are the only nodes chosen as forwarders for this network.

not need to cover nodes $D$ and $E$, because they are adjacent to node $B$ that is in the sender's forwarder list. $C$ determines node $F$ as its forwarder node, which in turn determines node $E$ as a forwarder node. Node $F$ does not need to include node $E$ because this node is covered by a previous forwarder node (i.e., node $B$ that is in $A$'s forwarder list).

**Theorem 1.** *Given a graph $G(V, E)$, let $T_{DP}$ be a CDS of $G$ when applying the algorithm DP, and $T_{EDP}$ a CDS of $G$ when applying the algorithm EDP. Then $T_{EDP}$ is equivalent to $T_{DP}$.*

*Proof.* Nodes in the set $\mathcal{F}_{\mathsf{S_S}}$ and the set $\mathcal{F}_{\mathsf{S}}$ can be excluded without any implication besides reducing redundancy. Nodes in $\mathcal{Q}$ were already covered by some other forwarders in $\mathcal{F}_{\mathsf{S_S}}$, therefore they can be omitted. A node $n_k \in \mathcal{P}$ can be disregarded because node $n_k$ is covered by some other node $n_L$ in the set $\mathcal{I}$ (nodes with higher priority), i.e., all nodes in $N_1^k$ are also in $N_1^L$, or $n_k \in \mathcal{F}_L$ when there is a node $n_u$ exclusively covered by $n_k$ (i.e. $n_u \in N_1^k$ and $n_u$ is not a neighbor of any other node in $N_1^L$). Hence, all nodes covered by $T_{DP}$ are also covered by $T_{EDP}$. □

## 3.3 Applying EDP to AODV in the Context of Omni-Directional Antennas

This section addresses the application of EDP to the route discovery process in AODV (AODV-EDP) in the context of omni-directional antennas. Our neighbor protocol uses hello packets to disseminate the one-hop neighborhood, which creates a picture of its two-hop neighborhood at any given node in the network. A hello packet advertises the node's sequence number (*mySeqNum*), the identification of its known neighbors (*neighbors[]*), and the corresponding neighbors' sequence number (*neighSeqNum[]*). We have chosen a hello interval of $1.5s$. To reduce the number of broadcast messages, RREQ also advertise the one-hop neighborhood information, working as a hello message. This event reschedules any pending hello message.

To avoid pruning too many route requests in the presence of mobility and cross-traffic, we have chosen to implement the neighbor protocol as part of AODV. We extended the hello mechanism available in AODV to include the information about the one-hop neighborhood in hello messages, and we also rely on the AODV mechanisms for evaluating the link status to neighbors.

A route request (RREQ) works in a similar way as in AODV. The main difference being that only forwarders rebroadcast a broadcast packet. The source of a RREQ calculates its forwarder list using EDP, and broadcasts the packet. Upon receiving a route request, a forwarder that cannot respond to this request calculates its own forwarder list using the information provided in the RREQ packet (i.e., forwarder list, second to previous forwarder list,

and source node) and broadcast the packet after updating it with its own forwarder list. Eventually the request reaches a node with a route to the destination or the destination itself. It is expected that most of the replies will come from an intermediate node because of the two-hop neighborhood information.

Because of topology changes, nodes may not have correct two-hop neighborhood information, which may result in forwarding lists that do not cover all nodes in the neighborhood. However, this is not a major problem, because a node incorrectly excluded from the forwarder list also receives the request and can respond in the case it has a route to the destination.

### 3.3.1 Simulations and Performance Results

To compare AODV with EDP (AODV-EDP) against other protocols, we use traffic and mobility models similar to those previously reported by Perkins et al [46]. We implemented AODV-EDP using the Qualnet™ [1] network simulator, and compare it against AODV-DP (AODV with *Dominant Pruning*), AODV with no hello messages and with $2s$ hello timers, and OLSR. We have chosen AODV and OLSR because they represent some of the most referenced reactive and proactive unicast routing protocols for wireless ad hoc networks.

#### Simulation Parameters

The network is composed of $50$ nodes spread over an area of $1500m\ x\ 300m$. The radio model used is a $2Mbps$ IEEE $802.11$ device with a nominal transmission range of $280m$.

Initially nodes are placed uniformly over a grid. Nodes move according to the random waypoint model with velocities between $0$ and $20m/s$. Seven pause times are tested: $0s$ (always moving), $50s$, $100s$, $300s$, $500s$, $700s$, and $900s$.

For traffic sources we use $30$ source nodes transmitting $4\ packets/s$ of $512$ bytes, making it a total of $120$ data packets being injected into the network every second. Nodes begin transmitting at $50s$ plus an offset uniformly chosen over a $5s$ period to avoid synchronization in their initial transmission. Source and destination pairs are chosen uniformly among the nodes in the network. The simulation time is set to $600$ seconds, and identical mobility and traffic scenarios are used across protocols.

Experiments are repeated for 10 trials with different random number seeds. Results present a $95\%$ confidence interval. Each data point represents the mean over the 10 runs discarding the lowest and largest results (quantile of one).

Four performance metrics are evaluated:

- *Packet delivery ratio*, the ratio of the data packets delivered to the destination to those generated by the CBR sources.
- *Average end-to-end delay* for data packets, including all possible delays caused by route discovery latency, queuing at the interface, retransmission delays at the MAC layer, and propagation and transfer times.
- *Routing load*, the number of routing packets transmitted per data packet delivered to the destination, where each hop traversed by the packet is counted as one transmission.
- *MAC collisions*, the number of collisions detected at the MAC layer.

**Results**

We show that AODV-EDP outperforms the other protocols in most of the performance metrics. OLSR performs better than AODV-EDP in terms of routing load and the number of MAC collisions (a difference of about $10\%$ less collisions). However, we have to analyze these results together with the other metrics.

For example, a lower end-to-end delay might be resulting from a large sample of packets delivered through the shortest connections, and the total number of packets delivered could be a way smaller than for other protocols, giving us a wrong impression about the whole picture. Besides that, we do not expect improvements for all the metrics analyzed because there is always a trade-off between improvements in one aspect and losses in some others.

Figure 3.5 shows the packet delivery ratio. AODV-EDP presents an almost constant packet delivery ratio for all pause times. As the network becomes more static, the proactive approach of OLSR starts to payoff and it performs better than standard AODV, but AODV-EDP has a higher delivery ratio for all the pause times. AODV-DP shows that DP alone can improve AODV; however, it also shows that there is room for more improvement (i.e., there is some more redundancy that can be eliminated). OLSR performs better than AODV-DP for large pause times (after $500s$ pause time).

As pointed out by Perkins et al [46], the possibility of link failures is low with low mobility, but due to the node movement model (random way-point) nodes usually get clustered. This situation is responsible for congestion in those regions in the presence of high traffic. This causes the link layer to report link failures even though the nodes are relatively static and a physical link still exists between the nodes. This is observed on Figure 3.5, where

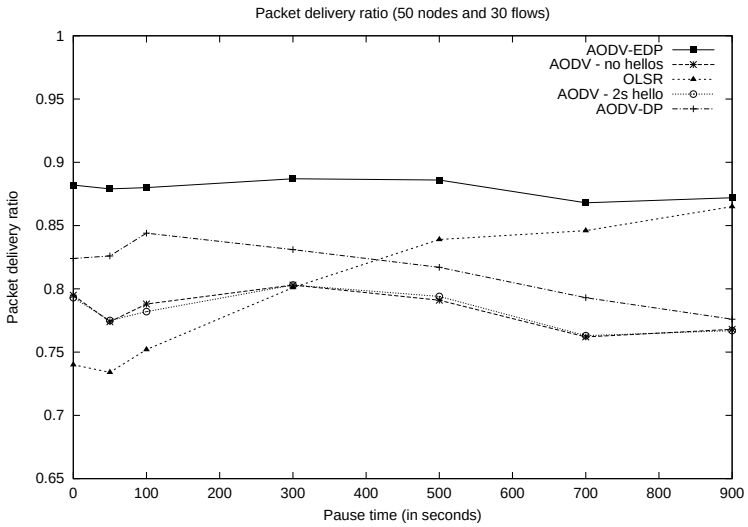



**Figure 3.5**: Packet delivery ratio for 50 nodes and 30 flows (120 packets/s)

we notice a decreasing on the packet delivery ratio for some larger pause times.

Figure 3.6 shows the average end-to-end delay. AODV-EDP presents an almost constant mean latency, and is always the best for all pause times. Together with the packet delivery ratio, these results show that besides delivering more packets AODV-EDP delivers them faster than the other protocols. AODV-DP again shows that DP alone improves AODV, but OLSR is still better than AODV-DP for large pause times. Clustering of nodes has a direct impact on the latency as well. Packets spend more time waiting on the queues, and usually need to be retransmitted due to increased congestion.

Figure 3.7 presents the routing load. As expected, AODV-EDP has a lower routing load in comparison to standard AODV, because it reduces the number of broadcast transmissions. As expected, AODV-DP reduces the control overhead compared to AODV, but not as much as AODV-EDP. OLSR has the lowest routing load, but at the same time it gets a comparable delivery ratio only when the network is more static. As mobility increases, OLSR does

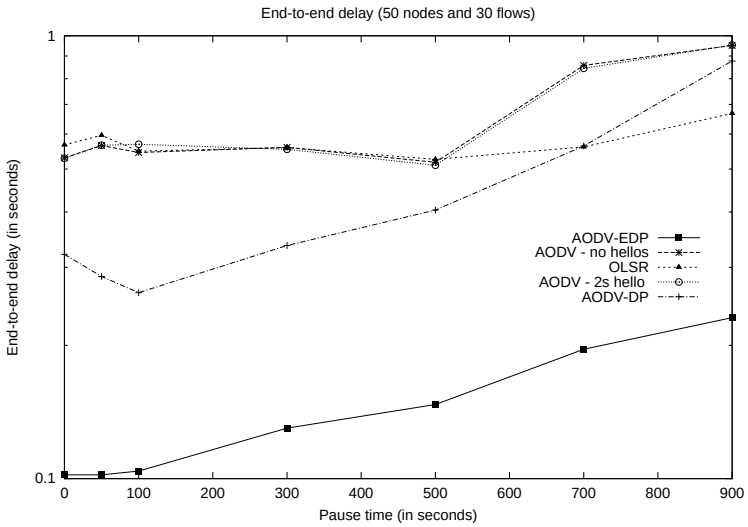



**Figure 3.6**: End-to-end delay for 50 nodes and 30 flows (120 packets/s)

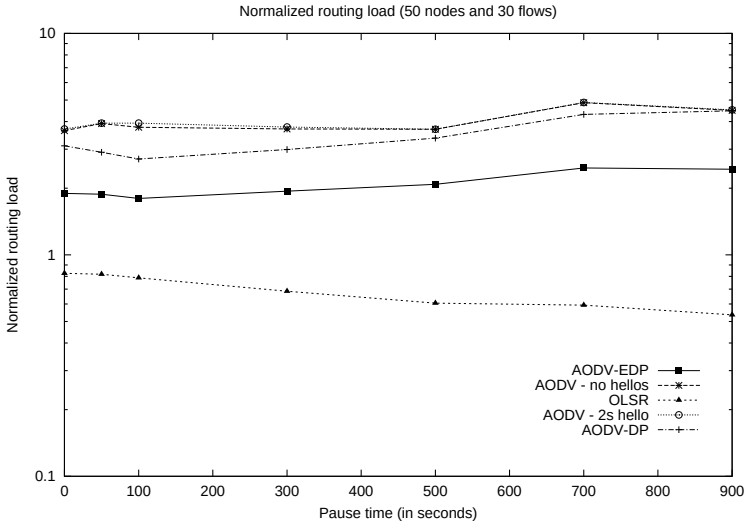



**Figure 3.7**: Routing load for 50 nodes and 30 flows (120 packets/s)

not deliver as many packets as AODV-EDP, and does not improve the end-to-end delay for any pause time. In another words, less control overhead does not translate in better performance for the upper layers.

Figure 3.8 shows the number of collisions at the MAC layer. The number of colli-

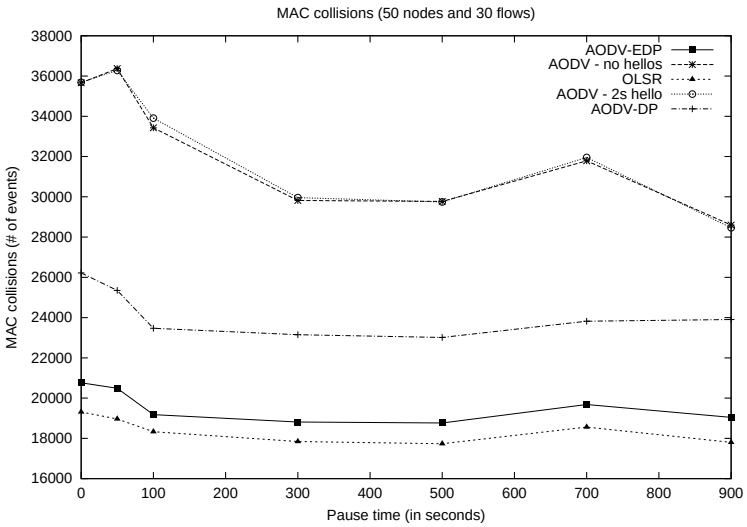



**Figure 3.8**: MAC collisions for 50 nodes and 30 flows (120 packets/s)

sions for standard AODV is noticeable larger than the other protocols, because a node always responds to the first received RREQ (if the TTL is valid, i.e., greater than zero). Because both AODV-EDP, AODV-DP, and OLSR reduce the number of necessary broadcasts, it translates in less collisions. OLSR produces slightly fewer collisions than AODV-EDP. However, these results when interpreted together with the packet delivery ratio and the end-to-end latency of both protocols indicate that AODV-EDP incurs a few more collisions because it delivers more packets.

Because the scenarios we have used to evaluate our approach differ from those presented in [37], and because we implemented our solution together with a neighbor and routing protocol, we do not know how our solution compares to TDP and PDP. The relation between the savings of pruning (too much, or too little) and the degree of broadcast redundancy achieved, can be different, depending on the physical environment under consideration. If we take into account that more packets being broadcasted translate into more contention and

collisions, we could have a different picture, depending on the number of broadcasts that are avoided.

## 3.4 Applying EDP to AODV in the Context of Directional Antennas

In addition to applying EDP to reduce the number of nodes that need to propagate RREQs transmitted on broadcast mode, information regarding prior routes to a destination is used to unicast RREQs to a region close to the intended destination, so that broadcast RREQs are postponed as much as possible and occur (if necessary) only close to the destination, rather than on a network-wide basis [52].

Directional antennas are assumed, which provide higher spatial reuse [29] [66] than omni-directional antennas for unicast transmission. An advantage of using directional antennas is that they allow a larger number of simultaneous transmissions compared to omni-directional antennas.

Figure 3.9 presents the pseudo-code for the modified RREQ. A route request (RREQ) is handled as follows:

- If the source of a RREQ does not have any previous knowledge about the route to the destination or is retrying the RREQ, it calculates its forwarder list using EDP, and broadcasts the packet (Lines 8, 9, and 14).

- On the other hand, if the source of a RREQ has knowledge about a recently expired route to the destination, and there is a valid route to the next hop towards the destination

**Figure 3.9**: RREQ Algorithm

```
   Data: n_i, destination D, F_S, F_Ss, U_i
   Result: Unicast the RREQ, or Broadcast the RREQ
   begin
1      if recently expired route to D and not retrying then
2          NextHop ⟵ previous_nextHop(D)
3          if validRoute(NextHop) then
4              result ⟵ Unicast
5          else
6              result ⟵ Broadcast
7      else
8          result ⟵ Broadcast
9      F_i ⟵ EDP(n_i, F_S, F_Ss, U_i)
10     Update RREQ packet with F_i
11     if result == Unicast then
12         Unicast the RREQ packet to NextHop
13     else
14         Broadcast the RREQ packet
   end
```

(Lines 2, 3, and 4), the node calculates the forwarder list using EDP (Line 9), but instead of broadcasting the RREQ packet, the node unicasts the packet to the last known next hop towards the destination (Line 12).

- Upon receiving a route request, a forwarder that cannot respond to this request calculates its own forwarder list using the information provided in the RREQ packet (i.e., forwarder list, second to previous forwarder list, and source node) and broadcasts or unicasts the packet (depending on which one of the two first cases apply) after updating it with its own forwarder list.

Eventually, the RREQ reaches a node with a route to the destination or the destination itself. Our approach attempts to reduce the number of collisions and the delay of the route discovery by unicasting a RREQ towards the region where the destination was previ-

ously located. The success of this approach depends on how fresh the previous known route to the destination is, and how fast the destination node is moving out of the previous known location. If an intermediate node has completely removed any route to the destination, the RREQ is then broadcasted. The intended effect is to postpone the broadcast of a RREQ to the region closest to the destination. In the case that the unicast approach fails, or there is no previous route to the destination, the source broadcasts by default.

Because of topology changes, nodes may not have correct two-hop neighborhood information, which may result in forwarding lists that do not cover all nodes in the neighborhood. However, this is not a major problem when the request is broadcasted, because a node incorrectly excluded from the forwarder list may also receive the request and is able to respond in the case it has a route to the destination.

### 3.4.1 Simulations and Performance Results

Qualnet™ [1] provides two models for directional antennas: *switched beam* with multiple patterns (circular array with 8 patterns), and *steerable* with multiple steerable patterns (triangular array with $4$ different beam widths). The antenna model is receiver side only due to the omni-directional MAC protocol. In our simulations we have used the *switched beam* model for all the simulations and routing protocols. The radio model used is a $2Mbps$ IEEE $802.11$ device. Terrain size and radio range are adjusted for each particular scenario.

Traffic sources are continuous bit rate (CBR). Only $512$-bytes data packets are used. The source-destination pairs are chosen randomly among the nodes in the network. Flows last in average for $50s$ (following an exponential distribution), unless otherwise mentioned.

Source nodes keep active flows during all simulation time (new destinations are randomly selected as needed). Nodes begin transmitting at $50s$ plus an offset uniformly chosen over a $5s$ period to avoid synchronization in their initial transmissions. The simulation time is set to $600$ seconds, and identical mobility and traffic scenarios are used for all protocols. Initially nodes are placed uniformly over a grid. Nodes move according to the random way-point model with velocities between $1$ and $20m/s$. Six pause times are tested: $0s$ (always moving), $50s$, $100s$, $300s$, $400s$, and $600s$.

Experiments are repeated for $10$ trials with different random-number seeds. Results present a $95\%$ confidence interval. Each data point represents the mean over the $10$ runs discarding the lowest and largest results (quantile of one).

**50-Node Scenario**

The network is composed of $50$ nodes spread over an area of $1500m\ x\ 300m$. The radio has a nominal transmission range of $250m$. The network is tested for three traffic models:

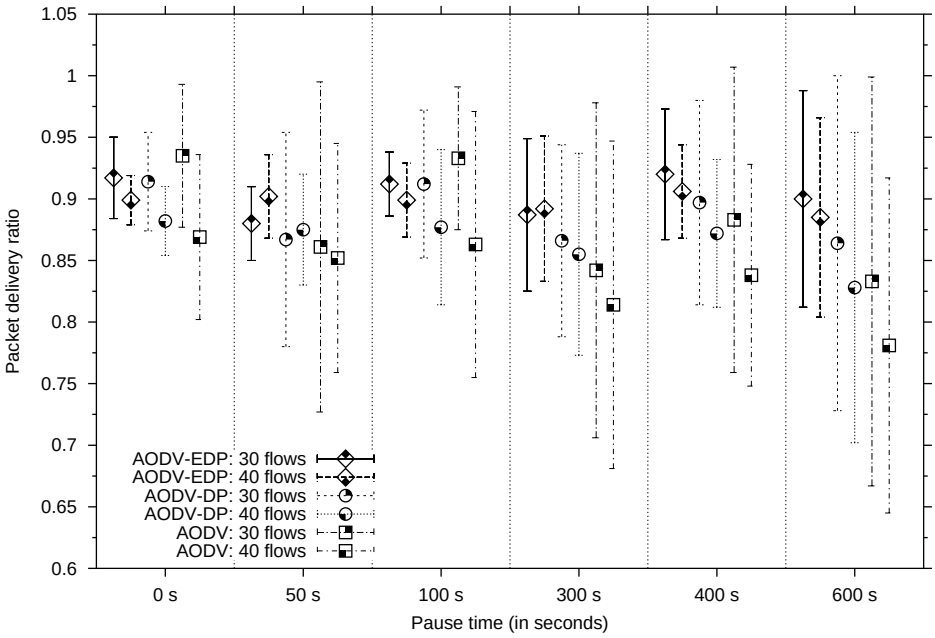
- $30$ source nodes transmitting $4\ packets/s$, each flow lasting in average $50s$ (exponential distribution).
- $40$ source nodes transmitting $3\ packets/s$ (flows of $50s$ as in the previous scenario).
- $30$ source nodes transmitting $4\ packets/s$, with very short flows (flows lasting in average $10s$ and $20s$).

In both scenarios, we have a total of 120 data packets being injected into the network every second. We show that, in all of the categories, AODV-EDP outperforms the other protocols. Figures 3.10 and 3.11 summarize the results for 30 and 40 flows, and Figures 3.12 and 3.13 summarize the results for 30 sources varying the flow duration.

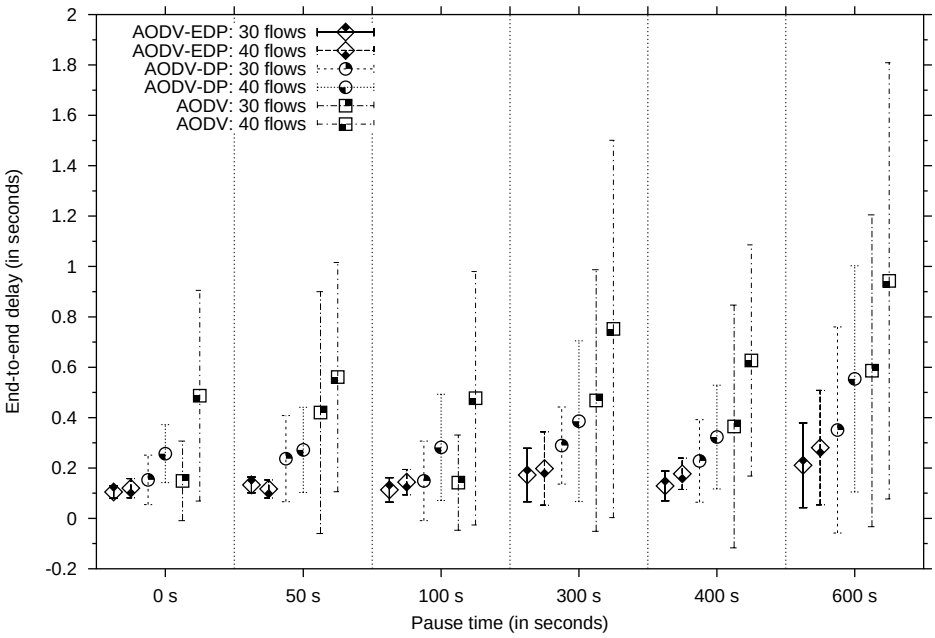
As pointed out by Perkins et al [46], the possibility of link failures is low with low mobility, but due to the node movement model (*random way-point*) nodes usually get clustered. This situation is responsible for congestion in those regions in the presence of high traffic. This causes the link layer to report link failures even though the nodes are relatively static and a physical link still exists between the nodes. This is observed on Figure 3.10(a), where we notice a decreasing on the packet delivery ratio for some larger pause times.

Figure 3.10(b) shows the average end-to-end delay for 30 flows. AODV-EDP presents an almost constant mean latency, and is always the best for all pause times. Together with the packet delivery ratio, these results show that besides delivering more packets for most of the pause times, AODV-EDP delivers them faster than the other protocols. AODV-DP again shows that DP alone improves AODV. Clustering of nodes has a direct impact on the latency as well. Packets spend more time waiting on the queues, and usually need to be retransmitted due to increased congestion.

Figure 3.11(a) presents the routing load for 30 flows. As expected, AODV-EDP has a lower routing load compared to standard AODV, because it reduces the number of broadcast transmissions. AODV-DP reduces the control overhead compared to AODV, but not as much as AODV-EDP.

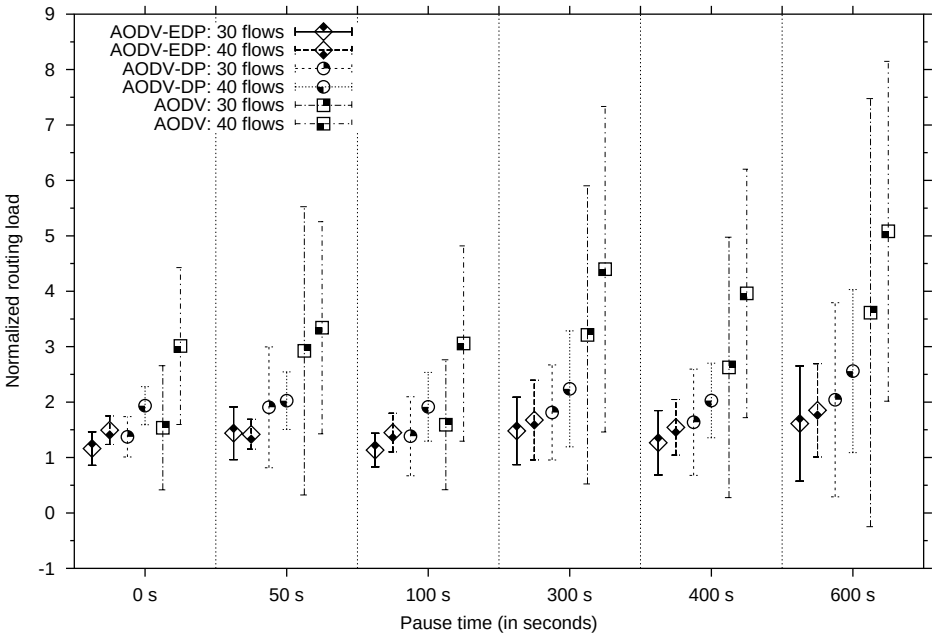



(a) Packet Delivery Ratio

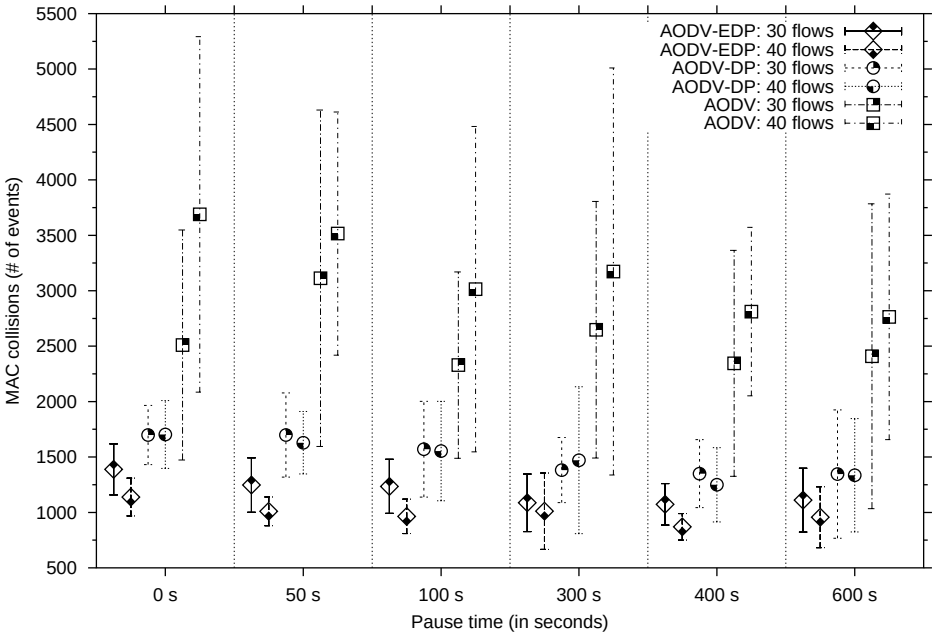



(b) End-to-End Delay

**Figure 3.10**: 50 Nodes, 30 and 40 flows: packet delivery ratio and end-to-end delay




(a) Control Overhead




(b) MAC Collisions

**Figure 3.11**: 50 Nodes, 30 and 40 flows: control overhead and MAC collisions

Figure 3.11(b) shows the number of collisions at the MAC layer for 30 flows. The number of collisions for standard AODV is noticeable larger than the other protocols, because a node always respond to the first received RREQ (if the TTL is valid, i.e., greater than zero). Because both AODV-EDP and AODV-DP reduce the number of necessary broadcasts, it translates in less collisions.

In this scenario we increase the number of flows but keep the same number of data packets being injected into the network (each source sends 3 packets/s). Figure 3.10(a) shows the packet delivery ratio. AODV-EDP presents an almost constant packet delivery ratio for all pause times, and it has a higher delivery ratio for all the pause times. The effect of clustering is noticeable on Figure 3.10(a). This result shows that by increasing the number of flows, more nodes in the network participate in active communications, what translates in more replies coming from intermediate nodes during the route discovery process. In these circumstances, it helps even more when a request can be unicasted instead of broadcasted.

Figure 3.10(b) shows the average end-to-end delay for 40 flows. AODV-EDP presents an almost constant mean latency, and is always the best for all pause times. In this scenario AODV-EDP again deliver more packets, and doing it faster than the two other variants. AODV-DP again shows that DP alone improves AODV. For all the pause times, AODV-DP presents less than half the latency produced by AODV. On its turn, AODV-EDP reduces even more the end-to-end delay, having almost all the time half the latency produced by AODV-DP. The impact of clustering of nodes in the latency of data packets is more noticeable only for AODV.

Figure 3.11(a) presents the routing load for $40$ flows. As expected, there is an increase in the routing load because there are more flows (and destinations) in the network. It is more noticeable the improvements introduced by both dominant pruning techniques, but AODV-EDP performs better for all pause times.

Figure 3.11(b) shows the number of collisions at the MAC layer for $40$ flows. Although a larger number of flows, for both AODV-DP and AODV-EDP we notice only a slightly difference (sometimes even less collisions) compared to the $30$ flows scenario. But AODV incurs on more collisions than on the previous scenario. In all situations AODV-EDP outperforms the two other variants.

In this set of simulations we play with the flow duration. At any given time, there are at least $30$ active flows, and every node in the network has a chance to be the source of at least one session . In fact, because we are dealing with flows of short duration, every node participates as a sender and as a receiver on several different sessions during the simulation time. Flows last in average $10s$ and $20s$ (exponential distribution). As mentioned before, flows start at $50s$ of simulation time with a jitter of $5s$. For each flow duration, simulations are run for the same number of trials as in the previous scenarios.

The results presented in Figures 3.12 and 3.13 show that DP alone improves the performance of AODV for all pause times and for all flows. But AODV-EDP performs better than the other two protocols in all situations, and it also presents the smallest variance among the three protocols. Both AODV-DP and AODV-EDP present an almost constant performance for all pause times. As expected, we notice again a great reduction on the control overhead due to the pruning of redundant broadcasts. But we also notice that AODV performs as well

as the other protocols regarding number of collisions in situations with large pause times and flows of $20s$.

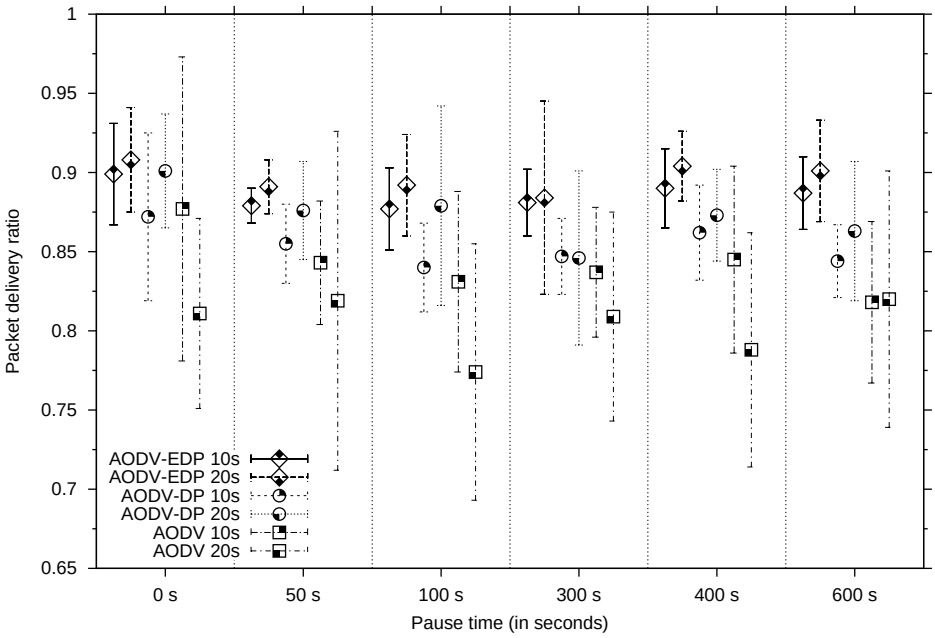
**100-Node Scenario**

The network is composed of $100$ nodes spread over an area of $2200m\ x\ 600m$. The radio has a nominal transmission range of $280m$. For traffic sources, we have two traffic models: $40$ source nodes transmitting $3\ packets/s$, and $60$ source nodes transmitting $2\ packets/s$. In both cases we have a total of $120$ data packets being injected into the network every second. We show that, in most of the categories, AODV-EDP outperforms the other protocols. For all the metrics evaluated, AODV-EDP presents the smallest variance.

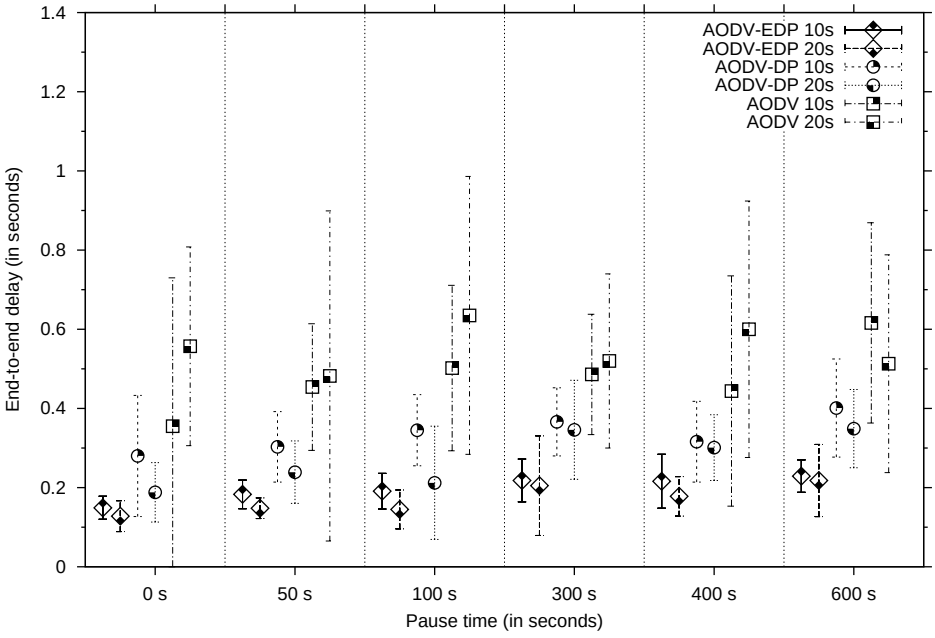
Figure 3.14(a) shows the packet delivery ratio for $40$ flows. AODV-EDP presents an almost constant packet delivery ratio for all pause times, as well as a higher delivery ratio for all pause times. AODV-DP performs worse than AODV specially in the high mobility scenarios, as the network gets more static the difference between AODV and AODV-DP becomes very small.

Figure 3.14(b) shows the average end-to-end delay for $40$ flows. AODV-EDP presents an almost constant mean latency, and is always the best for all pause times. Together with the packet delivery ratio, these results show that besides delivering more packets, AODV-EDP delivers them faster than the other protocols. Although AODV-DP performs better than AODV, AODV-DP delivers less packets than AODV.

Figure 3.15(a) presents the routing load for $40$ flows. As expected, AODV-EDP has a lower routing load in comparison to standard AODV, but the difference among the protocols

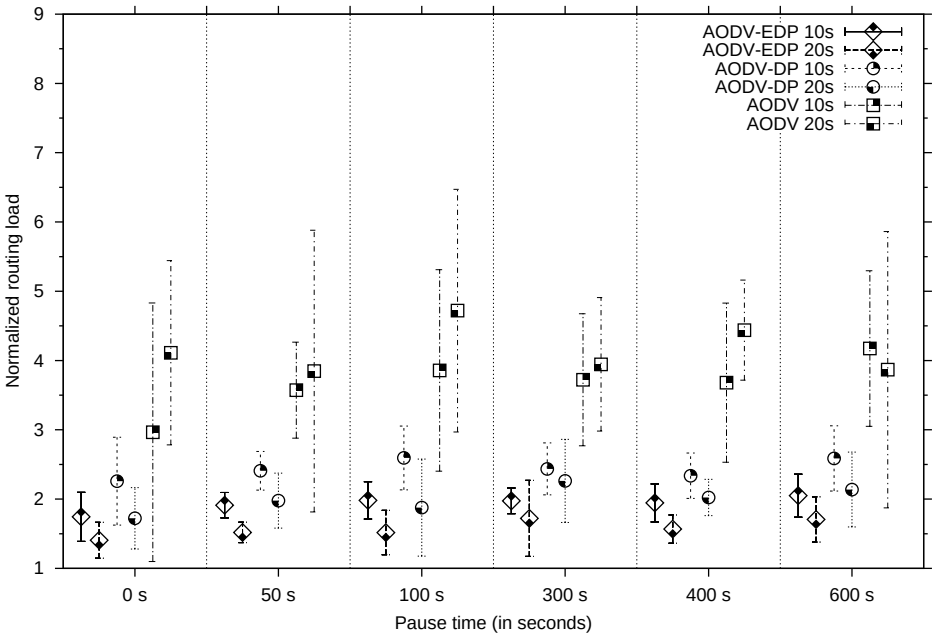



(a) Packet Delivery Ratio

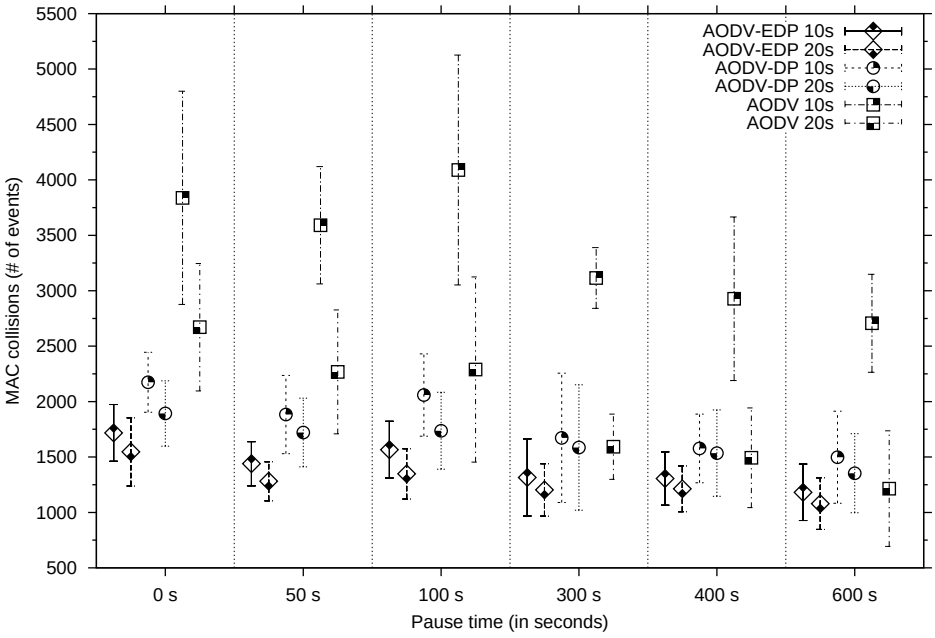



(b) End-to-End Delay

**Figure 3.12**: 50 Nodes, flows of $10s$ and $20s$: packet delivery ratio and end-to-end delay

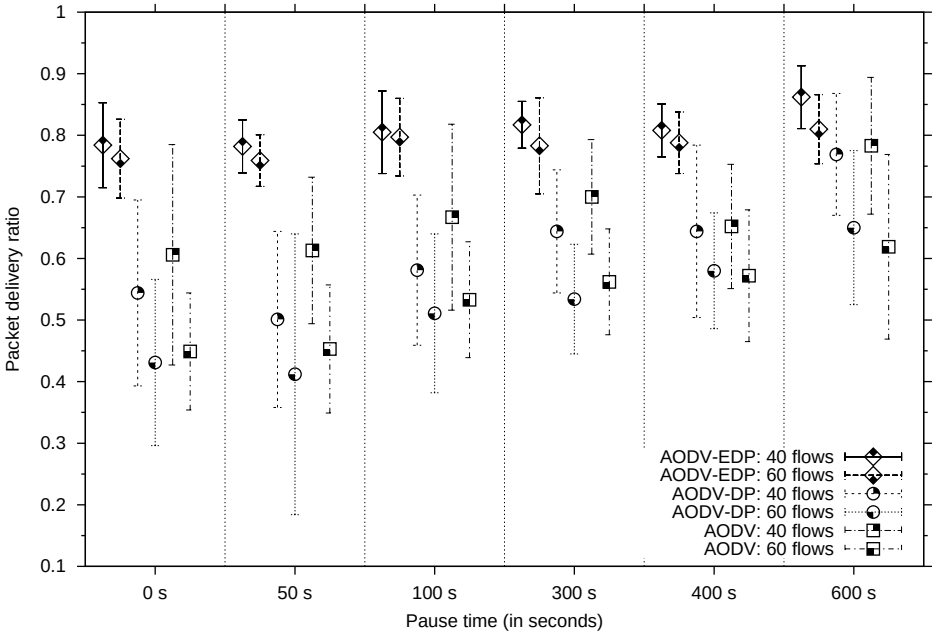



(a) Control Overhead

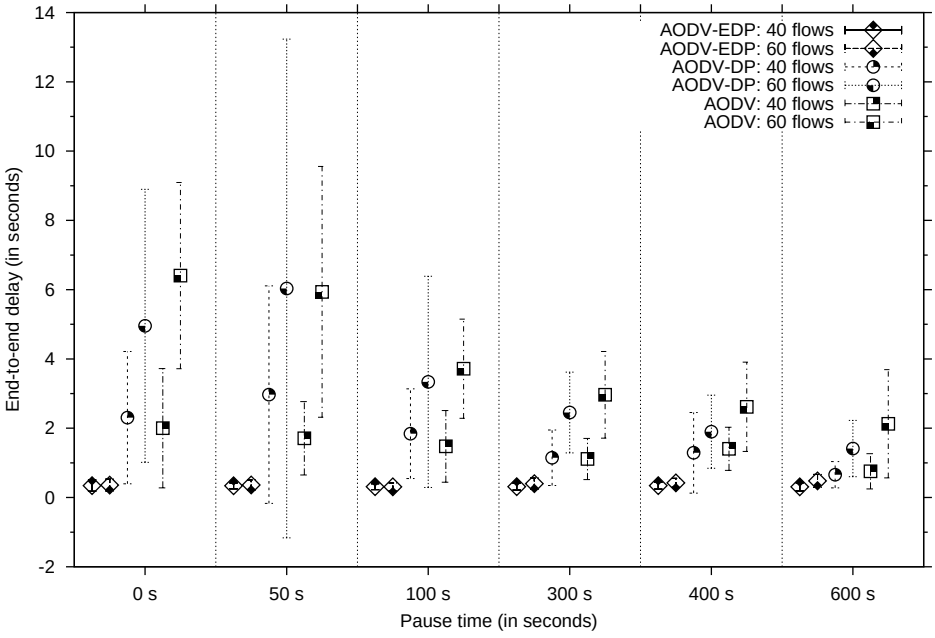



(b) MAC Collisions

**Figure 3.13**: 50 Nodes, flows of $10s$ and $20s$: control overhead and MAC collisions




(a) Packet Delivery Ratio




(b) End-to-End Delay

**Figure 3.14**: 100 Nodes, 40 and 60 flows: packet delivery ratio and end-to-end delay

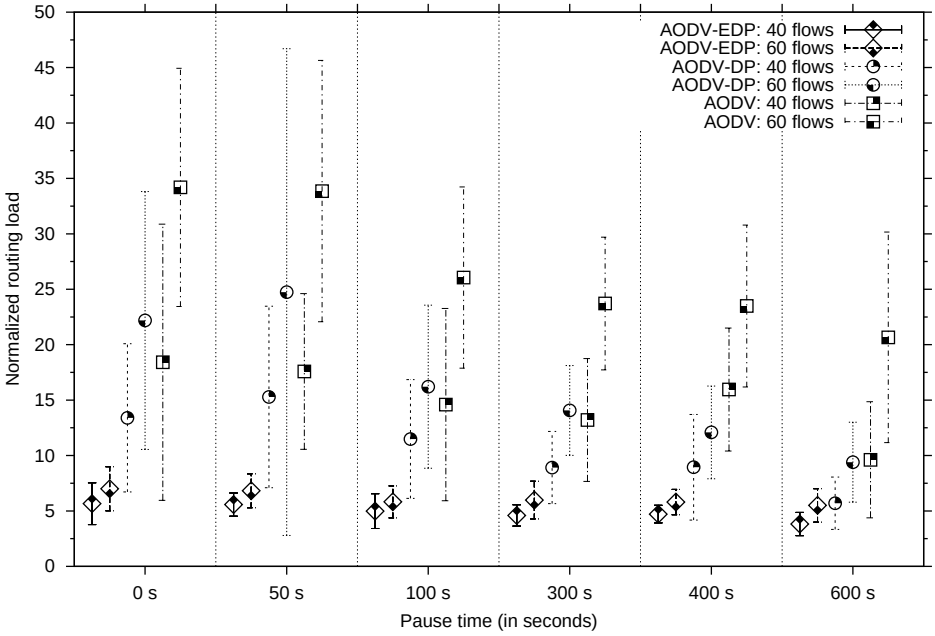
is a way larger than in the $50$ nodes scenario. AODV-DP reduces the control overhead compared to AODV, but not as much as AODV-EDP. AODV-DP shows that DP alone improves the control overhead, but it does not improve as much as EDP.

Figure 3.15(b) shows the number of collisions at the MAC layer for $40$ flows. As expected, the number of collisions for standard AODV is noticeable larger than the other protocols. AODV-EDP incurs $4$ to $5$ times less collisions than AODV for most of the pause times, and almost half of the collisions incurred by AODV-DP. AODV-EDP also presents the smallest variance, and an almost constant number of collisions for all pause times.

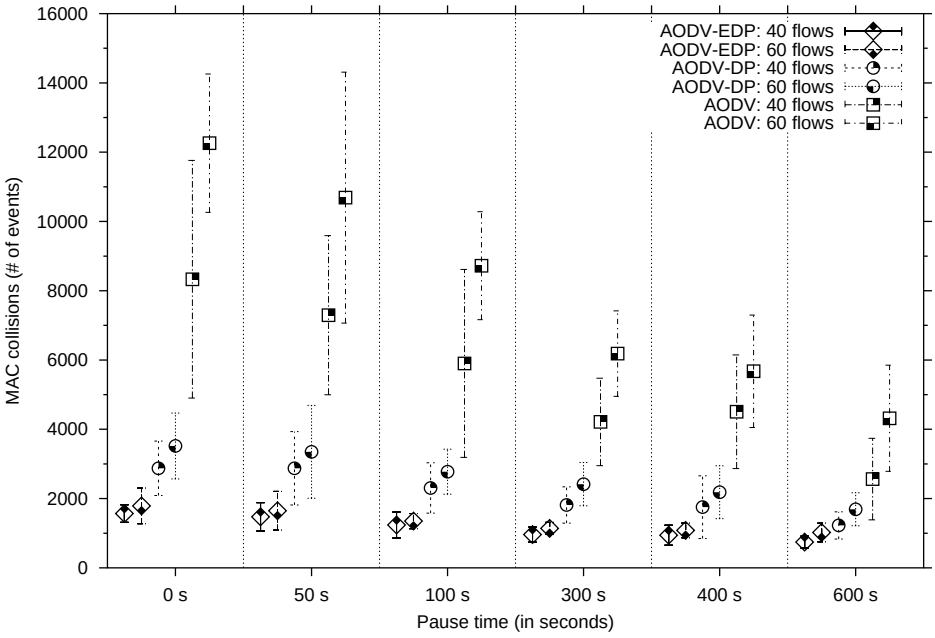
Figure 3.14(a) shows the packet delivery ratio for $60$ flows. AODV-EDP presents an almost constant packet delivery ratio for all pause times, as well as a higher delivery ratio. AODV-DP performs worse than AODV but the difference is smaller compared to the $40$ flows scenario.

Figure 3.14(b) shows the average end-to-end delay for $60$ flows. AODV-EDP presents an almost constant mean latency, and is always the best for all pause times. As in the previous scenarios, besides delivering more packets, AODV-EDP delivers them faster. AODV-DP performs better than AODV, but it also delivers slightly less packets than AODV. As expected, the latency increases compared to the $40$ flows scenario, but not as much for AODV-EDP.

Figure 3.15(a) presents the routing load for $60$ flows. AODV-DP performs better than AODV, specially for larger pause times. AODV-EDP is the best again, and when comparing the results against the $40$ flows scenario, we observe that only AODV-EDP does not increase the control overhead proportionally as observed in the two other protocols.




(a) Control Overhead




(b) MAC Collisions

**Figure 3.15**: 100 Nodes, 40 and 60 flows: control overhead and MAC collisions

Figure 3.15(b) shows the number of collisions at the MAC layer for $60$ flows. Compared to the $40$ flows scenario, AODV increases about $50\%$ the number of collisions, while both AODV-EDP and AODV-DP increase around $15\%$ the number of collisions. Both pruning techniques show to be effective on reducing redundant broadcasts, but EDP outperforms DP in all aspects.

## 3.5 Conclusions

We presented an enhanced dominant pruning approach that allows pruning redundant broadcasts even more than the conventional dominant pruning heuristic. Redundant broadcasts increase the number of packet collisions, and consequently delay the response for RREQs in the route discovery process. EDP is shown to reduce the number of broadcast transmissions when compared to standard DP. Because EDP requires the two-hop neighborhood to determine the forwarder list, we built a neighbor protocol as part of AODV. By making the neighbor protocol part of AODV, the result is a more accurate view of the local topology, and therefore more accurate is the determination of the forwarder list.

In the context of omni-directional antennas, AODV-EDP improves the packet delivery ratio for all the pause times tested in the $50$ nodes and $30$ flows scenario. The other protocols (standard AODV and OLSR) deliver fewer packets than AODV-EDP (the only exception is at $900s$ when OLSR has the same delivery ratio as AODV-EDP). The end-to-end delay is much better in AODV-EDP, and is less than half of the delays incurred by the other two protocols. The better delivery ratio and lower latency do not come for free, and AODV-EDP incurs more normalized routing load than OLSR, but less than standard AODV. The reduction

of broadcast replicas by AODV-EDP and OLSR translates into a lower number of collisions at the MAC layer.

In addition to applying EDP to reduce the number of nodes that need to propagate RREQs transmitted on broadcast mode, information regarding prior routes to a destination is used to unicast RREQs to a region close to the intended destination, so that broadcast RREQs are postponed as much as possible and occur (if necessary) only close to the destination, rather than on a network-wide basis. Directional antennas are assumed, which provide higher spatial reuse than omni-directional antennas for unicast transmission. In this context, we show through extensive simulation results that AODV-EDP improves the performance in all aspects (i.e., the four metrics chosen) for all the pause times in the 50-node and the 100-node scenarios. The other protocols (standard AODV and AODV-DP) deliver fewer packets than AODV-EDP. AODV-EDP not only delivers more packets, but it does it faster than the other protocols. AODV-EDP also presents the smallest variance among the protocols, and almost constant results for all the metrics considered in the simulations (with some exceptions because of clustering of nodes due to the mobility model).

# Chapter 4

# Improving On-Demand Routing with Two-Hop Connected Dominating Sets

We introduce the *three-hop horizon pruning* (THP) [51, 53, 54] algorithm to make broadcast operations more efficient in ad hoc networks using contention-based MAC protocols. THP builds a *two-hop connected dominating set* (TCDS) of the network, which is a set of nodes such that every node in the network is within *two* hops from some node in the dominating set. Efficiency of broadcast operations is attained by implementing forwarding schemes that take advantage of a TCDS. More specifically, every node provides its one-hop neighbors with a list specifying one or more tuples, each with the identifier of a one-hop neighbor and a bit indicating if that neighbor dominates *any* two-hop neighbor. To forward a broadcast packet, a node tries to obtain the smallest subset of *forwarders*, which are one-hop neighbors that use some of the node's two-hop neighbors to reach *any* node that is three hops away. After such a selection of forwarders, the node broadcasts its packet with a header specifying its forwarder

list, and each forwarder in turn repeats the process.

THP is the first heuristic to take into account three-hop information in the selection of relay nodes for the broadcasting of packets, while incurring signaling overhead that is much the same as that of heuristics based on two-hop information. THP is also the first neighbor-designated algorithm for computing a TCDS. The one-hop neighbor list and the *one-hop dominating list* communicated to a node by its one-hop neighbors provide the node with a three-hop horizon of how a broadcast message can be propagated to nodes that are three hops away, even though they are unknown.

When a broadcast protocol based on neighbor information is used it is possible to maintain fresh routes to all nodes within two hops, because every node has the two-hop neighborhood information. For example, in on-demand routing protocols (e.g., the *Ad-hoc On-demand Distance Vector Protocol* (AODV) [45]) it is not necessary to broadcast the *route request* (RREQ) packet to every node in the network: disseminating it to a TCDS of the network is enough.

THP is shown to improve the performance of networks with low mobility when it is used for broadcasting of route request (RREQ) messages in AODV. However, because THP relies on an accurate view of the two-hop neighborhood, high mobility can degrade its performance considerably.

To address this problem, we propose the *Three-hop Horizon Enhanced Pruning* (THEP). A *virtual radio range* (VR), shorter than the physical radio range (RR), is used for gathering information about the local neighborhood (i.e., two-hop neighborhood). Instead of using two different transmission powers as proposed by Wu and Dai [70], a single transmis-

sion power is used while still managing to have a *buffer zone* in which neighbors can move without compromising network connectivity. Having two transmission powers, $t_{min}$ and $t_{max}$ (with $t_{min} < t_{max}$), can incur additional interference compared to having just one transmission power $t < t_{max}$, because the transmit power of each node appears as interference noise degrading the *signal-to-noise ratio* (SNR) [9]. In general, the greater the transmit power the higher the interference to other nodes' transmissions and receptions.

Upon receiving a broadcast packet, the forwarder list in the packet header is analyzed together with the current information about the local neighborhood. This is done to find inconsistencies between the most up-to-date *one-hop dominating list* and the one used by the sender to compute the sender's forwarder list. Changes in the local topology may have impacted the *one-hop dominating list*. If that is the case, a node may decide to relay a broadcast packet even though it was not selected as a forwarder by the sender.

## 4.1 Three-Hop Horizon Pruning (THP)

The most efficient broadcasting algorithms that have been proposed to date prune unnecessary transmissions using two-hop topology information at each node. Each node selects a subset of one-hop neighbor nodes whose transmissions reach all its two-hop neighbor nodes. Because every nodes carries out the same type of pruning, a broadcast packet can potentially reach all network nodes using fewer transmissions, depending on the reliability of the MAC layer.

In DP, the forwarder list is a set of one-hop nodes such that all two-hop nodes are covered. The approach we use in the Three-Hop Horizon Pruning (THP) algorithm is to make

the pruning process in DP more efficient by using topology information three hops away from a given node, while incurring very limited additional signaling overhead in conveying such information.

The information about the two-hop neighborhood of a node can be disseminated by means of a *neighbor protocol* that is independent of the routing protocol, or by periodically advertising the one-hop neighbor list using HELLO messages as part of the routing protocol. Without loss of generality, let us assume that nodes use HELLO messages to advertise the one-hop neighbor lists of nodes.

Based on the one-hop neighbor lists from its one-hop neighbors, each node can determine which one-hop neighbor it can use to reach any two-hop neighbor. Hence, node $n_j$ could derive a *one-hop dominating list*, $D^j_{\text{1-hop}}$, by running standard DP over the two-hop neighborhood as if node $n_j$ were the source (for notation refer to the List of Notations).

In addition to informing its one-hop neighbors about its one-hop neighbor list, node $n_j$ also communicates its *one-hop dominating list* $D^j_{\text{1-hop}}$ to its one-hop neighbors. To reduce the space required for this additional information, the *one-hop dominating list* is encoded in a bit-map format. Because a node lists all its one-hop neighbors in its HELLO message, and because the *one-hop dominating list* is a subset of the one-hop nodes (i.e., $D^j_{\text{1-hop}} \subset N^j_1$), it suffices to signal (i.e., one bit per node) which neighbors are *one-hop dominating nodes*.

The one-hop neighbor list and the *one-hop dominating list* communicated to a node by its one-hop neighbors provides the node with a *three-hop horizon* of how a broadcast message can be propagated to nodes that are three hops away, even though they are unknown. For node $n_i$, the set of all $D^j_{\text{1-hop}}$ for all $n_j \in N^i_1$, contain the set of two-hop nodes covering

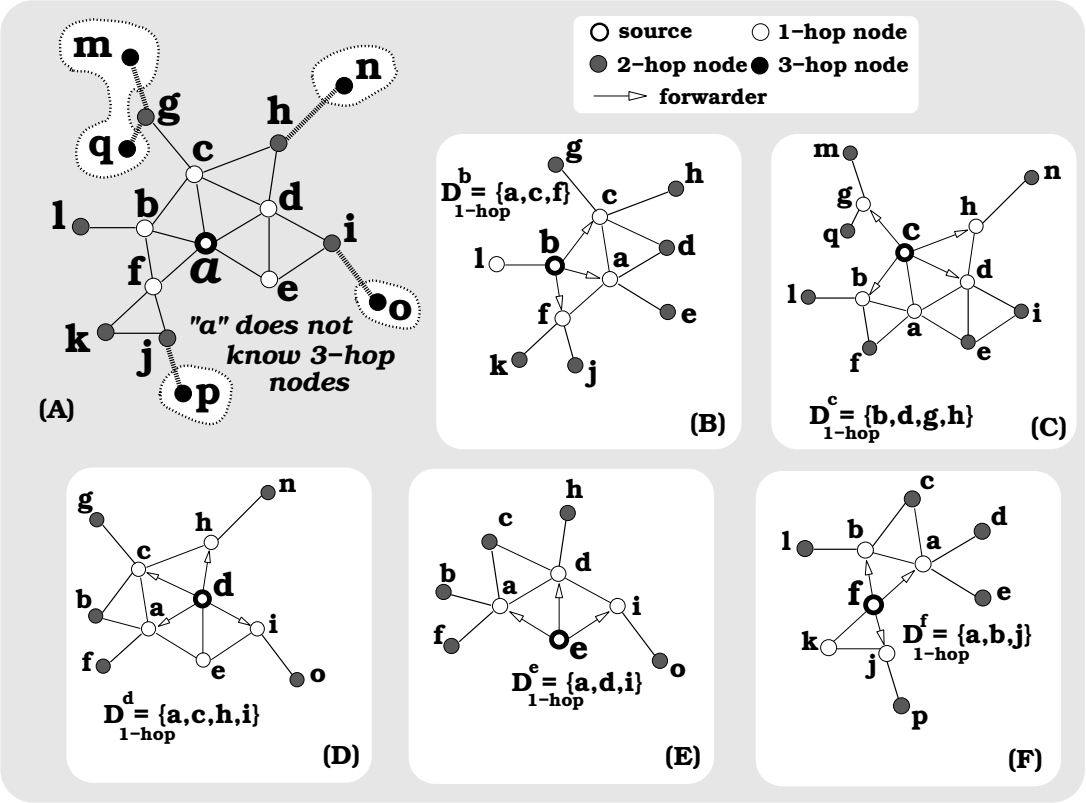



**Figure 4.1**: Network example: (A) Node $a$ knows its two-hop neighborhood, and the *one-hop dominating nodes* (i.e., $D_{\text{1-hop}}$ ) selected by each one-hop neighbor. A subset of nodes from $\bigcup_{n_j \in N_1^a} D^j_{\text{1-hop}}$ (i.e., $\{g, h, i, j\}$) cover all nodes in the three-hop neighborhood of node $a$ . (B–F) show the network from the point of view of each neighbor of node $a$, and how each $D_{\text{1-hop}}$ list is obtained via DP.

all three-hop nodes of node $n_i$. Figure 4.1 (A) shows an example network. Node $a$ knows its two-hop neighborhood, and also the *one-hop dominating list* advertised by each one-hop neighbor (along with the one-hop neighbor list). Figure 4.1 (B through F) show the network from the point of view of each one-hop neighbor of node $a$, and how they get to the *one-hop dominating list* (i.e., $D_{\text{1-hop}}$) by running DP. Excluding node $a$ itself and its one-hop neighbors, the list of nodes from all $D^j_{\text{1-hop}}$ for all $n_j \in N_1^a$ is reduced to $\{g, h, i, j\}$, and we can see that all three-hop nodes of node $a$ are covered by these set of nodes.

Instead of simply using the two-hop neighbor coverage as the main criteria for selecting forwarders as is done in standard DP, THP uses the advertised neighbor's *one-hop dominating list* (i.e., $D_{\text{1-hop}}$) to compute which one-hop neighbors have forwarders other than nodes in $N_1^i + n_i$ (i.e., nodes other than the node itself and its one-hop neighbors). Figure 4.3 presents the pseudo-code for THP (for notation refer to List of Notations). Let $\mathcal{C}$ be the list of nodes to be considered as candidates for forwarders. One-hop neighbors of the sender $S$ do not need to be taken into account (line 1), because the sender already did it. For all candidates to forwarders $n_k \in \mathcal{C}$, the list of nodes to be covered (i.e., set $\mathcal{U}[k]$) is built. From the list $D^k_{\text{1-hop}}$, only nodes that are not one-hop neighbors of the current node, $n_i$, and are not node $n_i$ itself, are included in the list $\mathcal{U}[k]$ (lines 2 through 6). The set to be covered, $\mathcal{U}$, is composed of all subsets $\mathcal{U}[k]$ for all nodes $n_k \in \mathcal{C}$. Nodes in $\mathcal{U}[k]$ that are covered (i.e., in another subset of $\mathcal{U}$ or a neighbor of some node in $\mathcal{C}$) by another node in $\mathcal{C}$ can be eliminated (lines 7 through 12, and Figure 4.2). For all candidates $n_k \in \mathcal{C}$ and for every node $n_m \in \mathcal{U}[k]$, the algorithm checks if there is another candidate to forwarder $n_l \in \mathcal{C}$ such that node $n_m$ is a neighbor of node $n_l$. If this is the case, then node $n_m$ can be removed from the set covered by node $n_k$ (i.e., $\mathcal{U}[k]$). In other words, if there is some candidate $n_l$ that is neighboring a node $n_m$ (which may or not be in $\mathcal{U}[l]$) that is in the set to be covered by candidate node $n_k$, then node $n_m$ does not need to be covered by node $n_k$, given that node $n_l$ being a neighbor of node $n_m$ did choose it as *one-hop dominating node* or has another neighbor covering the nodes covered by node $n_m$. In case node $n_l$ did not choose node $n_m$ as a *one-hop dominating node*, it may be the case that node $n_l$ has another neighbor(s) covering the nodes advertised by node $n_m$, or all neighbors of node $n_m$ are also neighbors of node $n_l$. If the set $\mathcal{U}[k]$ becomes empty, then

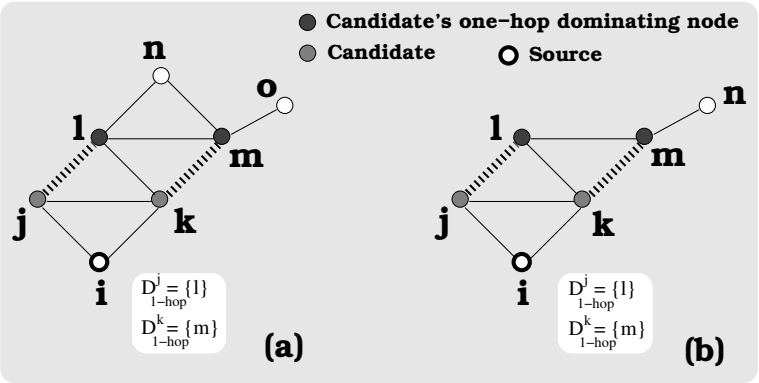



**Figure 4.2**: Node $k$ does not choose node $l$ as a forwarder (standard DP, for the hello message). In Case (a), another node (i.e., neighbor $m$) covers all nodes covered by $l$ (excluding those nodes covered by $k$), plus node $o$. In Case (b), all nodes covered by $l$ are one-hop neighbors of node $k$. In any case, it is safe to remove $l$ from $\mathcal{U}[j]$, because node $k$ covers all nodes covered by $l$, or has other neighbor(s) covering the nodes covered by $l$.

node $n_k$ is no longer a candidate to forwarder, and can be removed from the set $\mathcal{C}$ (lines 11 and 12). One restriction when eliminating redundancy from the set $\mathcal{U}$, is that a node $n_k$ must have all its nodes in the set $\mathcal{U}[k]$ checked before proceeding to the next node in the set $\mathcal{C}$. After all nodes in $\mathcal{C}$ are processed, the nodes remaining in the set $\mathcal{C}$ are selected as forwarders.

The following theorem proves that THP forms a TCDS in a connected network.

**Theorem 2.** *Given a connected graph $G(V, E)$, the node subset $N^{'}$, computed using the THP algorithm, forms a TCDS of $G$.*

*Proof.* By the definition of a *one-hop dominating set*, for any node $n_k$ in the network, the set $D^k_{\text{1-hop}}$ is a subset of nodes of $N^k_1$ such that all nodes in $N^k_2$ are covered. First, we consider the set of forwarders defined by the source, $n_i$, and then from the initial set of forwarders, $\mathcal{F}_i$, we show how the TCDS is constructed. For the source node $n_i$, the list of candidates to forwarders, $\mathcal{C}$, include all the one-hop neighbors of node $n_i$ (i.e., $N^i_1$). Because $n_i$ is the source, $S = \emptyset$. The set $\mathcal{U} = \sum_{j \in N^i_1} \mathcal{U}[j]$ cover all three-hop nodes of node $n_i$, because it

includes all the nodes covering the two-hop neighborhood of all neighbors of node $n_i$ (i.e., $\forall\ n_j \in N_1^i$, node $n_i$ knows $D^j_{\text{1-hop}}$). A node $n_k \in \mathcal{U}[j]$, such that node $n_k \in N_1^l$ for node $n_l \in \mathcal{C}$ ($n_l \neq n_j$), can be excluded from $\mathcal{U}[j]$, because node $n_k$ is covered by node $n_l$, which is another valid candidate to forwarder. This assertion holds given that all nodes in $\mathcal{U}[j]$ are processed before proceeding to the remaining nodes in $\mathcal{C}$ (i.e., for any node $n_j \in \mathcal{C}$, check this condition for all nodes in $\mathcal{U}[j]$, before proceeding to the next node $n_l \in \mathcal{C}$). Hence, the nodes in $\mathcal{U}$ cover all two-hop and three-hop nodes of node $n_i$. The set of forwarders, $\mathcal{F}_i$, is a subset of nodes in the set $\mathcal{C}$, such that all nodes in $\mathcal{U}$ are covered. On their turn, nodes $\{n_{j_1}, n_{j_2}, ..., n_{j_m}\} \in \mathcal{F}_i$ compute their sets $\mathcal{C}$, excluding the sender (i.e., $S = n_i$), and the one-hop neighbors shared with the sender ($N_1^j \cap N_1^i$), because these nodes are already considered by node $n_i$ when deriving the set $\mathcal{F}_i$. Nodes $\{n_{j_1}, n_{j_2}, ..., n_{j_m}\} \in \mathcal{F}_i$ derive their list of forwarders, i.e., $\{\mathcal{F}_{j_1}, \mathcal{F}_{j_2}, ..., \mathcal{F}_{j_m}\}$ (which can be an empty list in case no candidates lead to three-hop nodes). Each individual set in $\{\mathcal{F}_{j_1}, \mathcal{F}_{j_2}, ..., \mathcal{F}_{j_m}\}$ cover the three-hop neighborhood of nodes $\{n_{j_1}, n_{j_2}, ..., n_{j_m}\}$ respectively. Given that the set of nodes $\{n_{j_1}, n_{j_2}, ..., n_{j_m}\}$ cover the three-hop nodes of node $n_i$, the joint sets $\{\mathcal{F}_{j_1}, \mathcal{F}_{j_2}, ..., \mathcal{F}_{j_m}\}$ cover the four-hop nodes of node $n_i$. Therefore, the set of forwarders chosen subsequently cover all nodes $d + 3$ hops away from the source, where $d$ is the distance from the forwarder to the source. Because a forwarder is selected by a previous forwarder, or by the source, the set of forwarders is connected. Furthermore, because a forwarder checks for neighbors that reach three-hop nodes, it is guaranteed that, whenever there is at least one three-hop node, a forwarder is selected among the forwarder's one-hop neighbors. Because the selection process ends when no more three-hop nodes can be reached from a forwarder, it is guaranteed that any node in the network

is at most two hops from a forwarder. □

**Figure 4.3**: THP

**Data**: $n_i$ (any given node), $S$ (sender), $D^k_{\text{1-hop}}$ for all $k \in N^i_1$
**Result**: $\mathcal{F}_i$, the forwarder list
**begin**

```
1   C ⟵ N_1^i − N_1^S
    /* Select neighbors with one-hop dominating nodes
       other than one-hop neighbors and the node
       itself                                          */
2   for n_k ∈ C do
3       U[k] ⟵ ∅
4       for n_l ∈ D^k_1-hop do
5           if n_l ∉ (N_1^i + n_i) then
6               U[k] ⟵ U[k] + {n_l}
    /* Exclude candidates covered by another
       candidate in C                                  */
7   for n_k ∈ C do
8       for n_m ∈ U[k] do
9           if ∃ (n_l ≠ n_k) ∈ C | n_m ∈ N_1^l then
10              U[k] ⟵ U[k] − n_m
11              if U[k] == ∅ then
12                  C ⟵ C − n_k
    /* For every node n_k ∈ C, and for every n_m ∈ U[k],
       there is no other n_l ∈ C such that n_m ∈ U[l];
       therefore, all nodes in C are forwarders.       */
13  F_i ⟵ C
14  return F_i
end
```

### 4.1.1 Example of THP Operation

Figure 4.4 depicts an example of applying THP to compute a TCDS, having node $a$ as the source. First, lets consider the *one-hop dominating lists* announced by the neighbors of node $a$: $D^k_{\text{1-hop}} = \{g, h, o, p\}$, $D^p_{\text{1-hop}} = \{a, b, h, k\}$, $D^s_{\text{1-hop}} = \{a, p, b\}$, $D^o_{\text{1-hop}} = \{a, k, r\}$,

and $D^r_{\text{1-hop}} = \{a, o, w\}$. Because node $a$ is the source, all its one hop neighbors are candidates to be forwarders. We have that $\mathcal{U}[k] = \{h, g\}$, $\mathcal{U}[o] = \emptyset$, $\mathcal{U}[p] = \{h, b\}$, $\mathcal{U}[r] = \{w\}$, and $\mathcal{U}[s] = \{b\}$. Node $o$ is not a candidate, because it does not provide *one-hop dominating nodes* other than one-hop neighbors of node $a$, or node $a$ itself. In other words, node $o$ has no two-hop neighbors other than those reachable through node $a$'s neighbors or node $a$ itself. Therefore, there is no use to forward the packet toward node $o$. After excluding candidates covered by another candidate to be a forwarder (considering nodes in $\mathcal{C}$ are processed in alphabetical order), we obtain: $\mathcal{U}[k] = \{g\}$, $\mathcal{U}[p] = \emptyset$, $\mathcal{U}[r] = \{w\}$, and $\mathcal{U}[s] = \{b\}$. Notice that node $h$ is not listed in any of the $\mathcal{U}$'s list. However, there is at least one node (node $k$) in $\mathcal{C}$ covering node $h$, because a node $n_m$ can only be removed from list $\mathcal{U}[k]$ if there is another valid candidate $n_l$ covering node $n_m$ (i.e., $n_m \in N^l_1$). The source's forwarder list is then defined as $\mathcal{F}_a = \{k, r, s\}$. Now we look at each node in $\mathcal{F}_a$. Node $k$ has $\mathcal{C} = \{g, h\}$, $\mathcal{U}[g] = \emptyset$, and $\mathcal{U}[h] = \{i\}$. Note that both $p$ and $o$ are excluded from the set $\mathcal{C}$, because they are also neighbors of the sender. Based on that, we have that $\mathcal{F}_k = \{h\}$. Node $s$ has $\mathcal{C} = \{b\}$, and $\mathcal{U}[b] = \{l, t\}$, which gives us $\mathcal{F}_s = \{b\}$. Node $r$ has $\mathcal{C} = \{v, w\}$, $\mathcal{U}[v] = \emptyset$, and $\mathcal{U}[w] = \emptyset$; hence, it has no forwarders. Node $h$ has no forwarders, because the only candidate, node $i$, has no node in $D^i_{\text{1-hop}} = \{h\}$ other than node $h$ itself (i.e., $\mathcal{U}[i] = \emptyset$). For node $b$, we have a similar situation, where both candidates, nodes $l$ and $t$, lead to no other three-hop neighbor (i.e., $\mathcal{U}[l] = \mathcal{U}[t] = \emptyset$); hence, it has no forwarder.

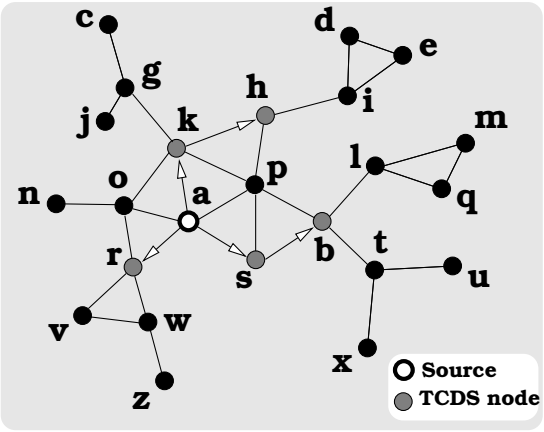



**Figure 4.4**: Building a TCDS using THP. Every *dominated* node is at most two hops from a *dominating* node

### 4.1.2 Efficacy of THP

This section evaluates the efficacy with which THP operates relative to other heuristics, when a TCDS is preferred over a CDS. We compare THP against the best-performing heuristics reported to date, namely DP, TDP, EDP, MPR, CC-I (with 2-hop neighborhood information, 2-hop routing history, and node-degrees as priority values), and an approximation to the MCDS problem. With the exception of MCDS (which is used as a lower bound for comparison purpose), all the other algorithms require the information about the two-hop neighborhood of nodes.

There is a clear trade-off between efficiency and reliability; that is, fewer nodes broadcasting reduces contention and collision of packets, but it may also reduce the chances of all nodes in the network receiving the broadcast packet. Therefore, the reliability of the MAC protocol is expected to affect the performance of any such broadcasting algorithms. To focus on the efficiency of the heuristics themselves, we use a customized simulator and

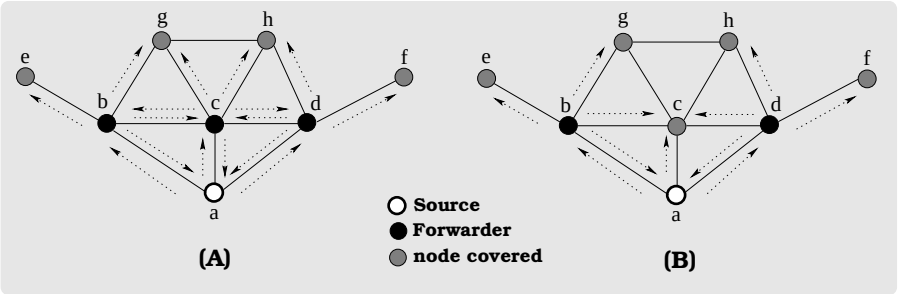



**Figure 4.5**: DP (a) versus MPR (b)

assume an *ideal* MAC protocol with which no collisions can occur. This is the same approach adopted in [15, 37, 68, 69] to compare the efficacy of heuristics.

DP is a distributed algorithm that determines a set cover based on the knowledge of the two-hop neighborhood. DP uses the *greedy set cover* (GSC) algorithm to compute the forwarder list of a packet. GSC recursively chooses one-hop neighbors that cover the most two-hop neighbors, repeating the process until all two-hop neighbors are covered.

Like DP, MPR also applies GSC in the selection of dominating nodes. However, MPR first chooses as forwarders those candidates that have exclusive coverage of some two-hop neighbor, and only then apply GSC over the remaining nodes. Fig. 4.5 shows an example illustrating the benefits of this approach over standard DP. With standard DP, node $a$ in Fig. 4.5 can start choosing *any* one of its neighbors, because they all have the same coverage area (i.e., two nodes, excluding node $a$ and the one-hop neighbors of $a$ ). Therefore, with standard DP node $a$ could choose node $c$ as a forwarder first, and in this case it would be forced to select its other two neighbors, because node $b$ is the only neighbor covering node $e$, and node $d$ is the only neighbor covering node $f$. In the MPR approach, nodes $b$ and $d$ must be selected first, and in this case node $c$ is not chosen, because nodes $b$ and $d$ cover all the two-hop nodes.

TDP [37] requires that the two-hop neighborhood of the sender be piggy-backed in the header of the packet. This information reduces the size of the two-hop neighbor set that needs to be covered by the forwarders. The header size increases proportionally to the number of nodes in the two-hop neighborhood, which may become a problem in dense networks. PDP [37] enhances DP by eliminating the two-hop nodes advertised by a neighbor shared by both the sender and the receiver (forwarder). EDP [50] requires the *second-to-previous* (STP) forwarder list in addition to the forwarder list, reducing the number of forwarders compared to DP.

In CC-I (dynamic), a node $n_i$ does not broadcast the packet if for any two neighbors $n_j$ and $n_k$, there is a path connecting them via several intermediate nodes with either higher priority values (e.g., node degree, node IDs) than node $n_i$, or with visited node status (i.e., the $h$ most recently visited nodes are included in the packet header).

Because the MCDS problem is an NP-complete problem, we use an approximation algorithm as a lower bound for the MCDS problem when comparing against the other algorithms. The algorithm used is based on the solution provided by Guha and Khuller [22]. This algorithm runs in polynomial time and achieves an approximation factor of $O(H(d))$, where $d$ is the maximum degree, and $H(d)$ is the $d^{th}$ harmonic number (i.e., $H(d) = \sum_{i=1}^{d} 1/i$). Nevertheless, this algorithm is not suitable for wireless ad-hoc networks, because it requires the knowledge of the whole network topology. The approximation algorithm used in [37] is not a good approximation because it uses a scanning rule that fails in some circumstances according to Guha and Khuller [22].

For the simulations, we vary the network size (i.e., number of nodes and terrain

**Table 4.1**: Terrain Size (in meters)

| # of nodes | Configuration 1 | Configuration 2 |
|---|---|---|
| 20 | 499 x 499 | 400 x 400 |
| 30 | 612 x 612 | 489 x 489 |
| 40 | 707 x 707 | 565 x 565 |
| 50 | 790 x 790 | 632 x 632 |
| 60 | 866 x 866 | 692 x 692 |
| 70 | 935 x 935 | 748 x 748 |
| 80 | 999 x 999 | 800 x 800 |
| 90 | 1060 x 1060 | 848 x 848 |
| 100 | 1118 x 1118 | 894 x 894 |
| 110 | 1172 x 1172 | 938 x 938 |
| 120 | 1224 x 1224 | 979 x 979 |
| 130 | 1274 x 1274 | 1019 x 1019 |
| 140 | 1322 x 1322 | 1058 x 1058 |
| 150 | 1369 x 1369 | 1095 x 1095 |
| 160 | 1414 x 1414 | 1131 x 1131 |
| 170 | 1457 x 1457 | 1166 x 1166 |
| 180 | 1500 x 1500 | 1200 x 1200 |
| 190 | 1541 x 1541 | 1232 x 1232 |
| 200 | 1581 x 1581 | 1264 x 1264 |

size) and measure the total number of forwarders for flooding the whole network. For each configuration (i.e., number of nodes and terrain size) we obtain the value for the metrics for $500$ arbitrary networks (nodes are randomly placed over the terrain, and connectivity is tested to ensure that the network is connected). Results represent the average over the $500$ different networks. The network size is varied from $20$ nodes to $200$ nodes. For the same number of nodes, we vary the terrain size according to two configurations so that we can test the algorithms for different node density (see Table 4.1). Configuration $1$ has a node density of $80$ nodes$/km^2$, and Configuration $2$ has $125$ nodes$/km^2$. For both configurations the radio range is set to $250m$; consequently we have that nodes in Configuration $2$ have, in average, larger node degree than nodes in Configuration $1$.

**Table 4.2**: *THP using DP* versus *THP using MPR*: number of forwarders (average $\pm$ standard deviation)

| # of nodes | THP using DP (average $\pm$ std) | THP using MPR (average $\pm$ std) |
|---|---|---|
| 20 | $2.102 \pm 0.066$ | $2.094 \pm 0.066$ |
| 30 | $4.356 \pm 0.085$ | $4.324 \pm 0.084$ |
| 40 | $6.978 \pm 0.096$ | $6.932 \pm 0.096$ |
| 50 | $9.928 \pm 0.118$ | $9.888 \pm 0.116$ |
| 60 | $12.884 \pm 0.127$ | $12.814 \pm 0.128$ |
| 70 | $16.226 \pm 0.146$ | $16.15 \pm 0.145$ |
| 80 | $19.474 \pm 0.154$ | $19.4 \pm 0.153$ |
| 90 | $22.706 \pm 0.165$ | $22.578 \pm 0.166$ |
| 100 | $26.3 \pm 0.187$ | $26.198 \pm 0.184$ |
| 110 | $30.104 \pm 0.188$ | $30.022 \pm 0.186$ |
| 120 | $33.576 \pm 0.21$ | $33.426 \pm 0.209$ |
| 130 | $37.468 \pm 0.212$ | $37.344 \pm 0.209$ |
| 140 | $40.882 \pm 0.23$ | $40.664 \pm 0.228$ |
| 150 | $44.654 \pm 0.23$ | $44.516 \pm 0.23$ |
| 160 | $48.978 \pm 0.249$ | $48.826 \pm 0.248$ |
| 170 | $52.438 \pm 0.249$ | $52.278 \pm 0.25$ |
| 180 | $56.156 \pm 0.260$ | $55.98 \pm 0.259$ |
| 190 | $60.584 \pm 0.291$ | $60.306 \pm 0.293$ |
| 200 | $64.256 \pm 0.304$ | $64.014 \pm 0.304$ |

Because THP prunes over the *one-hop dominating lists* advertised by the one-hop neighbors, computing the *one-hop dominating lists* using MPR instead of DP does not add much power to THP. Table 4.2 shows the results for THP (Configuration 1) based on DP and based on MPR. As we can see, there is a marginal difference between *THP with DP* and *THP with MPR*.

**Configuration** 1

Figure 4.6 presents the total number of forwarders for the six broadcasting algorithms. Because in THP nodes are at most two-hop away from a node in the TCDS, we

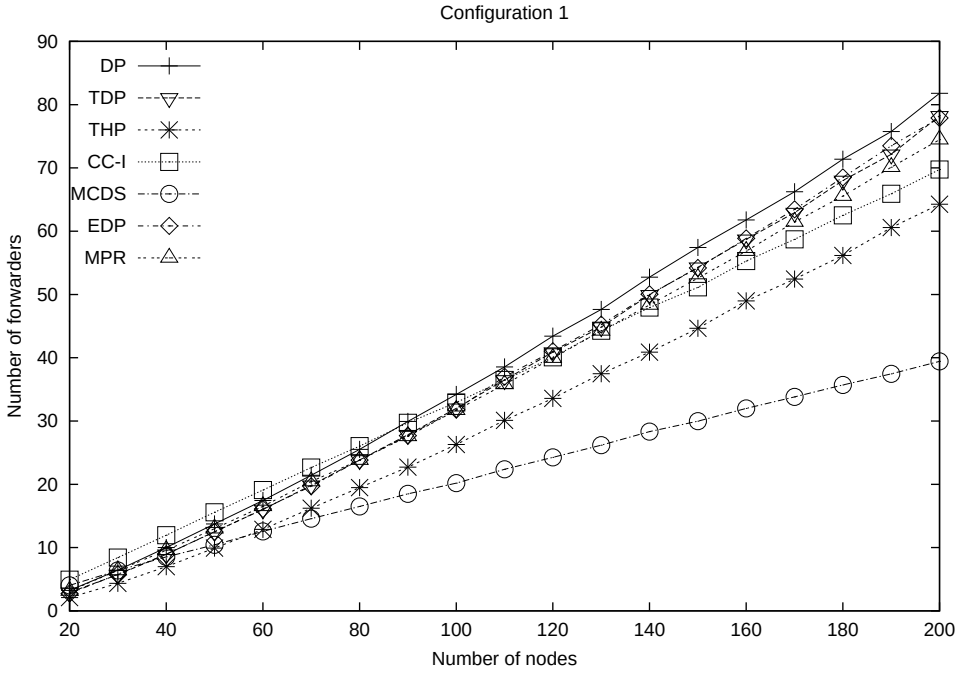



**Figure 4.6**: Configuration 1: average number of forwarders varying the number of nodes

have situations (e.g., small network sizes, 20 to 50 nodes) where THP produces a TCDS with smaller number of nodes than a CDS in MCDS. Anyway, MCDS is used as the reference to the best possible results for calculating the DS (but not feasible because we do not want to require the nodes in the wireless network to keep fresh information about the whole network topology). As expected, TDP improves DP for all network sizes, but it is more noticeable for larger networks (i.e., 100 nodes or more). EDP and TDP present similar results, with TDP performing slightly better for some network sizes. MPR performs better than DP, TDP, and EDP, for networks larger than 140 nodes. We notice that CC-I starts performing better than TDP only for larger networks (i.e., 140 nodes or more). In all circumstances, THP outperforms the other distributed approaches.

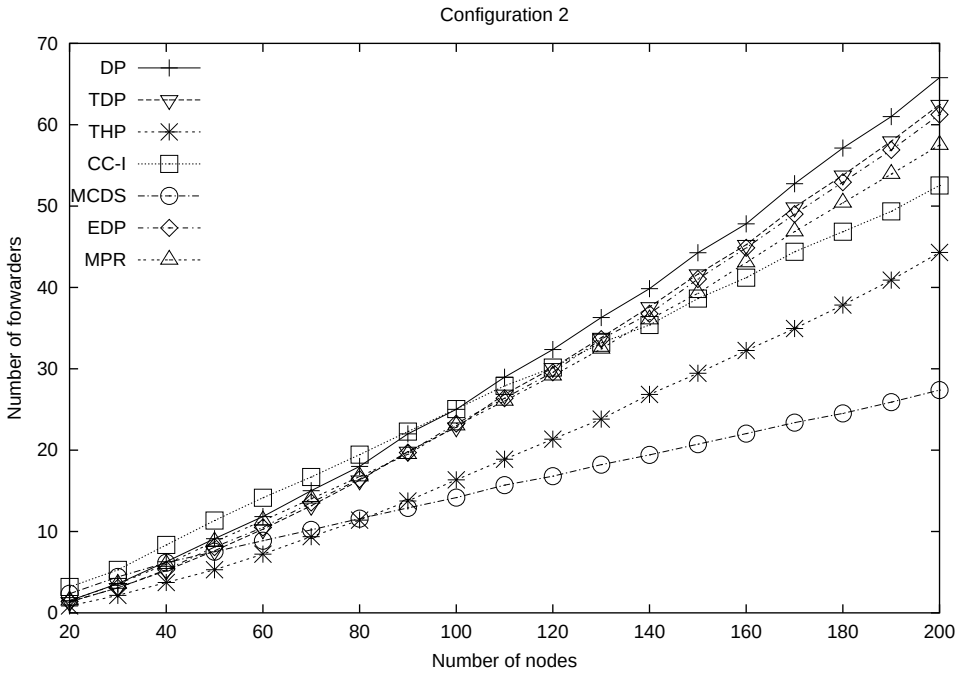



**Figure 4.7**: Configuration 2: average number of forwarders varying the number of nodes

**Configuration** 2

The networks in Configuration 2 are denser (i.e., larger node degree) than the networks in Configuration 1. Figure 4.7 shows the average number of forwarders for all broadcast algorithms. The difference between DP and TDP is more noticeable, because the networks are denser it pays off to have the two-hop neighborhood of the sender (i.e., in TDP) when calculating the set to be covered. EDP and TDP present similar results, but unlike in the previous configuration, EDP performs better for networks with more than 130 nodes. TDP performs better than CC-I for networks smaller than 120 nodes. MPR starts performing better than DP, TDP, and EDP, for networks larger than 130 nodes. But the difference between MPR and the other DP variants is more noticeable compared to the previous configuration. For all network

sizes, THP performs better than the other distributed broadcast algorithms. We also notice that THP performs better than MCDS for networks with 70 or fewer nodes. Once again, this particular behavior takes place because THP builds a TCDS instead of a CDS, and fewer nodes exist in the TCDS than in the CDS, especially for dense an small networks. The difference between THP and CC-I is more accentuated than in Configuration 1 for all the network sizes tested. This shows that the performance improvements attained with THP increases as the network gets denser.

### 4.1.3 Using THP for Route Discovery

THP can be applied to any type of broadcast operation that can take advantage of TCDS. One such operation is the dissemination of route requests (RREQ) in the route discovery process of on-demand routing protocols. For the purpose of discovering a route to a destination, it suffices that the RREQ reaches those nodes with a route to the desired destination. There are two cases to consider in terms of how THP can be used in this context.

If routes to two-hop neighbors are maintained pro-actively, then a node that is one or two hops away from the destination can reply to the RREQ directly.

On the other hand, if routes to two-hop neighbors are not available pro-actively, then a RREQ can be propagated in a number of ways once it reaches a node that is two hops away from the destination. The RREQ can be relayed using the expanding ring search with TTL set to 2. Alternatively, a node can compute forwarders within the two-hop neighborhood using a dominating set technique different than THP (e.g., DP).

To study the impact of THP on the route discovery process, we implemented THP

as the basis for deciding which nodes should broadcast RREQ messages in the route discovery process of AODV. We named the resulting protocol AODV-THP, and implemented it in *Qualnet* [1]. To compare AODV-THP against AODV, we use traffic and mobility models similar to those previously reported for the performance of AODV [46].

To address reliability, we used two versions of AODV-THP. First, AODV-THP implements THP as described previously. Second, we increase the coverage requirement of DP when computing the *one-hop dominating list* advertised in the HELLO messages (i.e., $D_{\text{1-hop}}$ ). Instead of requiring *at least one* dominating node (forwarder) per two-hop neighbor, every two-hop neighbor is covered by *at least two* forwarders (except when just one one-hop neighbor covers a two-hop node). This increases the chances that a two-hop neighbor receives a RREQ. This second variant is referenced as AODV-THP *two-cover*. The two variants of AODV-THP and AODV are tested with HELLO messages sent at a rate of $1s$ and $2s$. For AODV, we also present results without the use of HELLO messages.

AODV-THP would certainly incur much less overhead if it worked over a MAC protocol that exchanged the neighbor and forwarder information that we assume is exchanged as part of the routing protocol itself.

Experiments are repeated for $10$ trials with different random-number seeds, traffic endpoints, and topologies. Topology and traffic patterns are fixed using off-line generated mobility and packet generation scripts. This means that all protocols are compared having identical node mobility and traffic demands. Each data point represents the average of the $10$ trials.

Four performance metrics are evaluated:

- *Packet delivery ratio*, the ratio of the data packets delivered to the destination to those generated by the CBR sources.

- *Average end-to-end delay* for data packets, including all possible delays caused by route discovery latency, queuing at the interface, retransmission delays at the MAC layer, and propagation and transfer times.

- *Normalized routing load*, the number of routing packets transmitted per data packet delivered to the destination, where each hop traversed by the packet is counted as one transmission.

- *MAC collisions*, the number of collisions detected at the MAC layer.

Table 4.3 presents the set of parameters used in the simulations. The network is composed of $50$ nodes spread over an area of $1500m\ x\ 300m$. The radio model used is a $2Mbps$ IEEE $802.11$ device with a nominal transmission range of $280m$. Traffic sources are continuous bit rate (CBR). Only $512$-bytes data packets are used. The source-destination pairs are chosen randomly among the nodes in the network. Flows last in average for $30s$ (following an exponential distribution). Source nodes keep active flows during all simulation time (new destinations are randomly selected as needed). During the simulation time, an average of $580$ flows are initiated, and at any given time there are at least $30$ active flows. Nodes begin transmitting at $50s$ plus an offset uniformly chosen over a $5s$ period to avoid synchronization in their initial transmissions. The simulation time is set to $600$ seconds, and identical mobility and traffic scenarios are used for all protocols. Nodes are placed uniformly over a grid initially. Nodes move according to the random way-point model with velocities between $1$ and $20m/s$.

**Table 4.3**: Set of parameters used in the simulations of AODV-THP

| | |
|---|---|
| Number of nodes | 50 |
| Terrain size | 1500m $X$ 300m |
| Data rate | 2 $Mbps$ |
| Radio range | $280m$ for standard THP, and $250m$ for enhanced THP |
| MAC protocol | IEEE 802.11 |
| Data traffic, packet size | CBR, packets of 512 bytes |
| Number of flows, and duration | 30 active flows, lasting in average for $30s$ (exponential distribution); in average 580 flows are created during the simulation time. |
| Mobility model | random way-point (velocities between 1 and $20m/s$) |
| Pause times | $0s$ (always moving), $50s$, $100s$, $300s$, $400s$, and $600s$ (static) |
| Simulation time | $600s$ |

Six pause times are tested: $0s$ (always moving), $50s$, $100s$, $300s$, $400s$, and $600s$.

Figure 4.8 presents the packet delivery ratio results. As expected, AODV-THP does not perform very well in scenarios with frequent topology changes. One of the main reasons is that it is more difficult to get an accurate view of the local topology when it changes more frequently. As we increase the rate of HELLO messages (i.e., AODV-THP: 1s hello), THP improves its performance because, even though we are introducing more broadcast transmissions, nodes respond to topology changes faster. AODV-THP with HELLOs sent every $1s$ starts performing better than AODV at $300s$ pause time. When we increase the DP coverage from one to two dominating nodes (i.e., AODV-THP: 2 cover), we observe that the extra redundancy benefits the protocol in all situations, but specially for the high mobility scenarios. Here we can see the trade-off that exists between efficiency and reliability, and its relation with redundancy in broadcast transmission. For low mobility scenarios, it pays off to take advantage of a more accurate view of the local topology when making decisions about which node

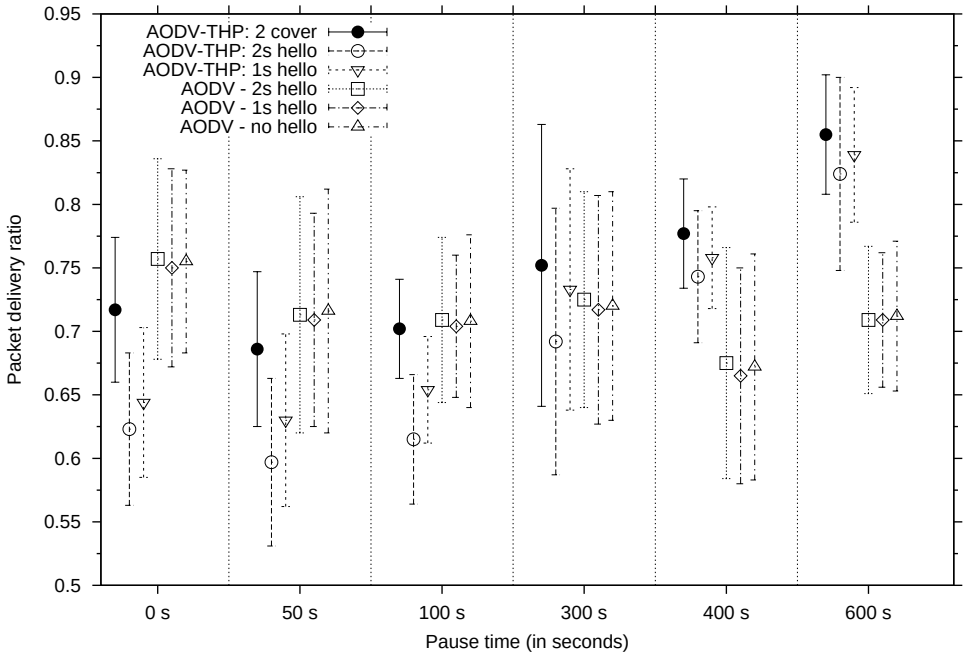



**Figure 4.8**: 50 nodes, 30 active flows (average of 580 total flows): Packet delivery ratio

should broadcast a packet. For high mobility scenarios, THP using a *2 cover* (i.e., AODV-THP: 2 cover) increases the delivery ratio by about 10% compared to the worst variant of THP.

Figure 4.9 presents the average end-to-end delay results. Given that approximately 580 flows are initiated during the simulation time, we observe that the large number of redundant broadcast transmissions (i.e., due to the route discovery process) affect the end-to-end delay in AODV. As shown previously, THP prunes more redundant broadcast transmissions than any other localized broadcast algorithm, and we can see here how it reflects in the overall performance of an on-demand routing protocol. The periodicity of HELLO messages reflect on the end-to-end delay as well. For THP, nodes keep a more accurate view of the local topol-

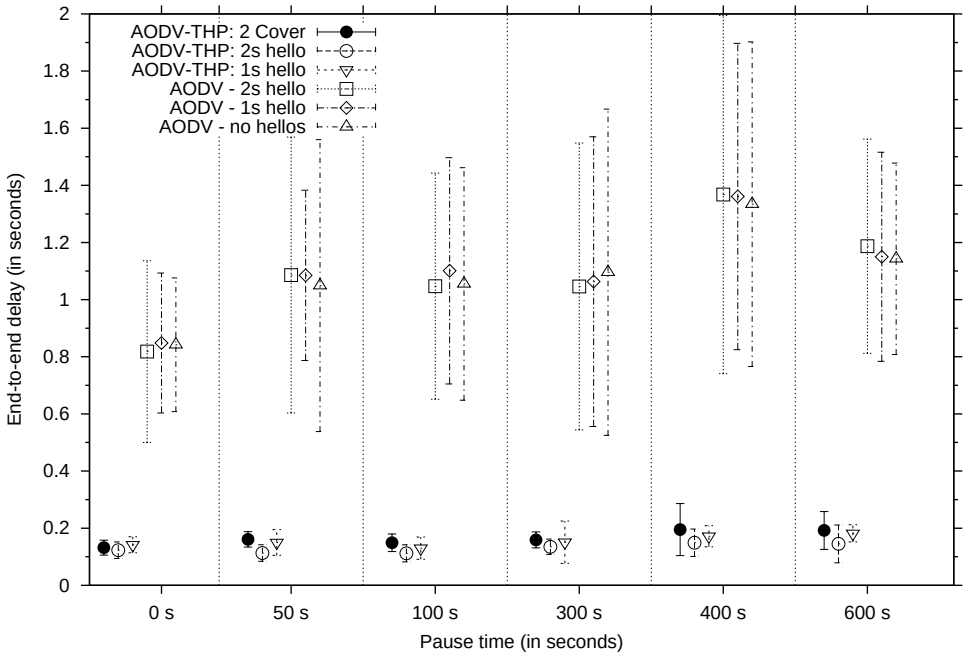



**Figure 4.9**: 50 nodes, 30 flows (average of 580 total flows): average end-to-end delay

ogy when HELLO messages are transmitted every $1s$. In this case, more packets are delivered, but they are also delivered faster. THP with *2 cover* redundancy (with hellos transmitted every $1s$ HELLO) presents a slightly larger average delay but it also delivers more packets for all pause-time values. For AODV, the frequency of HELLO transmissions do not affect much the end-to-end delay in such a scenario with a large number of flows. Together with the previous results for the delivery ratio, we can see that the reduction of redundant broadcast transmissions translate in a better and faster response to the route discovery process; consequently, more packets are delivered at a smaller cost.

Figure 4.10 shows the normalized routing load results (with respect to data packets delivered at the destination). All the THP variants present a much smaller overhead than

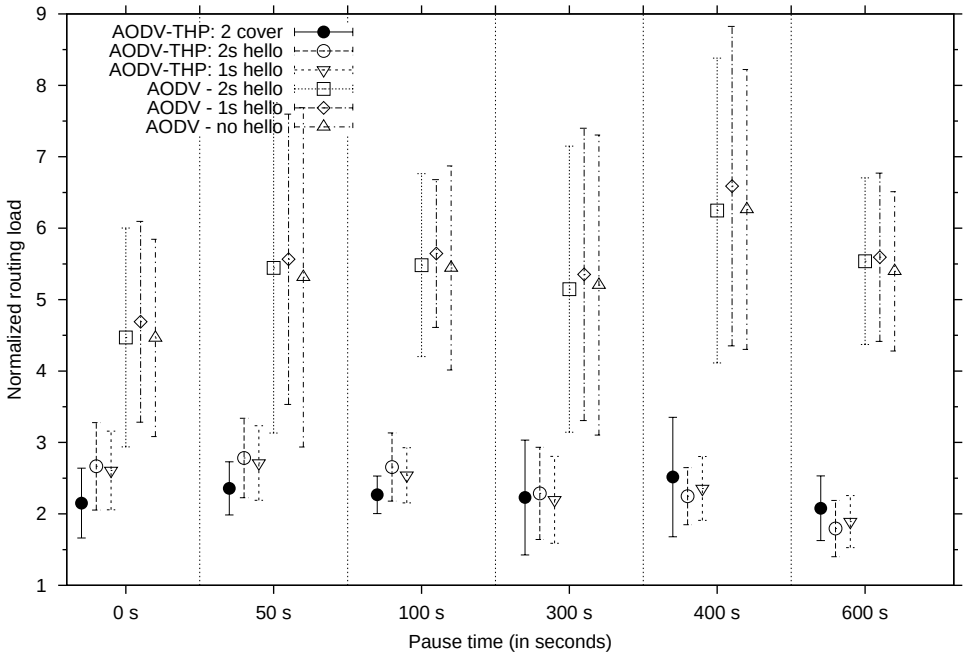



**Figure 4.10**: 50 nodes, 30 flows (average of 580 total flows): normalized routing load

AODV, because of the reduction on the number of redundant broadcast transmissions. As for the impact of the periodicity of HELLO messages, we observe slightly more control overhead in AODV when HELLO messages are sent every $1s$, compared to the two other variants. AODV without HELLOs, performs just slightly better than AODV with $2s$ HELLOs in terms of control overhead, delivery ratio, and end-to-end delay.

Figure 4.11 presents the results for the number of packet collisions. AODV with and without HELLOs attains similar results, showing that the increase in collisions is not due to the introduction of HELLO messages. The extra redundancy of RREQ transmissions is what results in more contention and collisions. As for AODV-THP, we observe that the periodicity of HELLO messages has a direct impact on the number of collisions, and that is

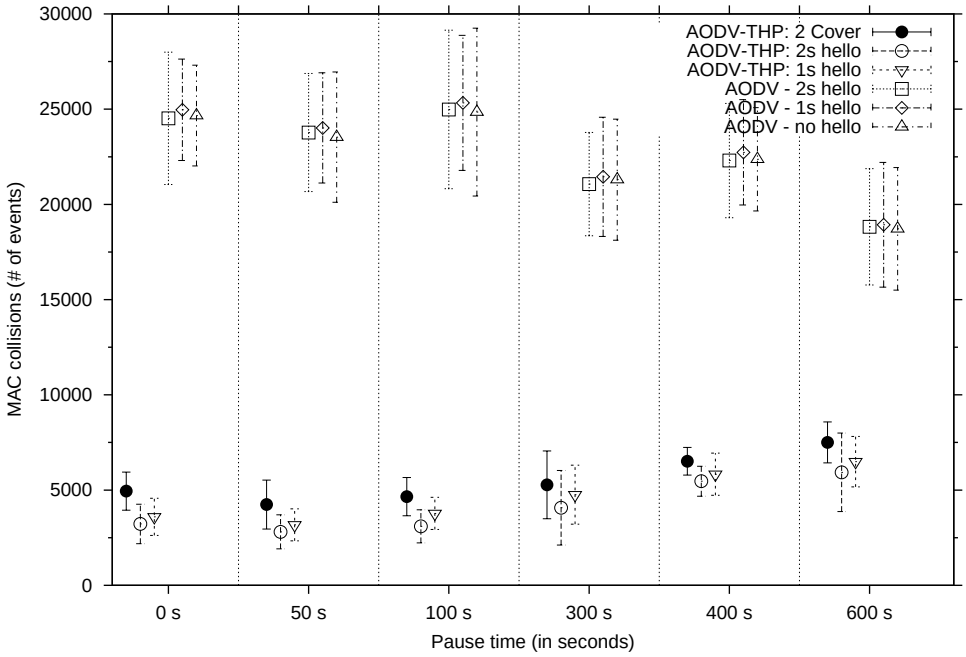



**Figure 4.11**: 50 nodes, 30 flows (average of 580 total flows): number of MAC collisions

because we reduce significantly the number of redundant broadcast, such that the introduction of any extra broadcast transmissions (i.e., HELLO messages) reflects in more contention and collisions in the network. Considering all the previous results, THP is shown to improve AODV performance in all aspects for scenarios with low mobility (i.e., pause time larger than $100s$).

## 4.2 Three-Hop Horizon Enhanced Pruning (THEP)

Because techniques such as THP rely on an accurate view of the two-hop neighborhood, it is expected not to perform well under high mobility. To tackle this problem, we propose two enhancements to THP, and name the new algorithm *Three-Hop Horizon Enhanced*

*Pruning* (THEP).

Neighbor information is maintained using a virtual radio range (shorter than the physical radio range): rather than using two different radio ranges (as in [70, 71]), we use neighbor location, and regard as neighbors only those nodes within virtual radio range.

To cope with changes in the local topology, information provided by the forwarder list and the freshest information about the local neighborhood are used to decide if the node should broadcast the packet even though it was not selected as a forwarder by the sender.

With this two enhancements, we expect to address the lack of reliability in the presence of high mobility. As in the work by Wu and Dai [70, 71], the gap between the virtual and physical ranges constitutes a buffer zone in which neighbors can move without incurring loss of connectivity. However, our approach applies just one transmission power, instead of two different transmission powers [70, 71]. Having two transmission powers, $t_{min}$ and $t_{max}$ (with $t_{min} < t_{max}$), can incur additional interference compared to having just one transmission power $t < t_{max}$, because the transmit power of each node appears as interference noise degrading the *signal-to-noise ratio* (SNR) [9]. In general, the greater the transmit power the higher the interference to other nodes' transmissions and receptions.

Because we need to know if a node is within virtual radio range, we can either use node location information (provided by GPS, for instance), or estimate the distance to the node based on the signal strength of the receiving packet [27]. In the first case, the information about the node location should be piggy-backed in the HELLO message along with the neighbor list. The second option is effective, and does not add as much complexity to the system as the first one. In any case, the exact location of the node is not needed; estimating if a node is within

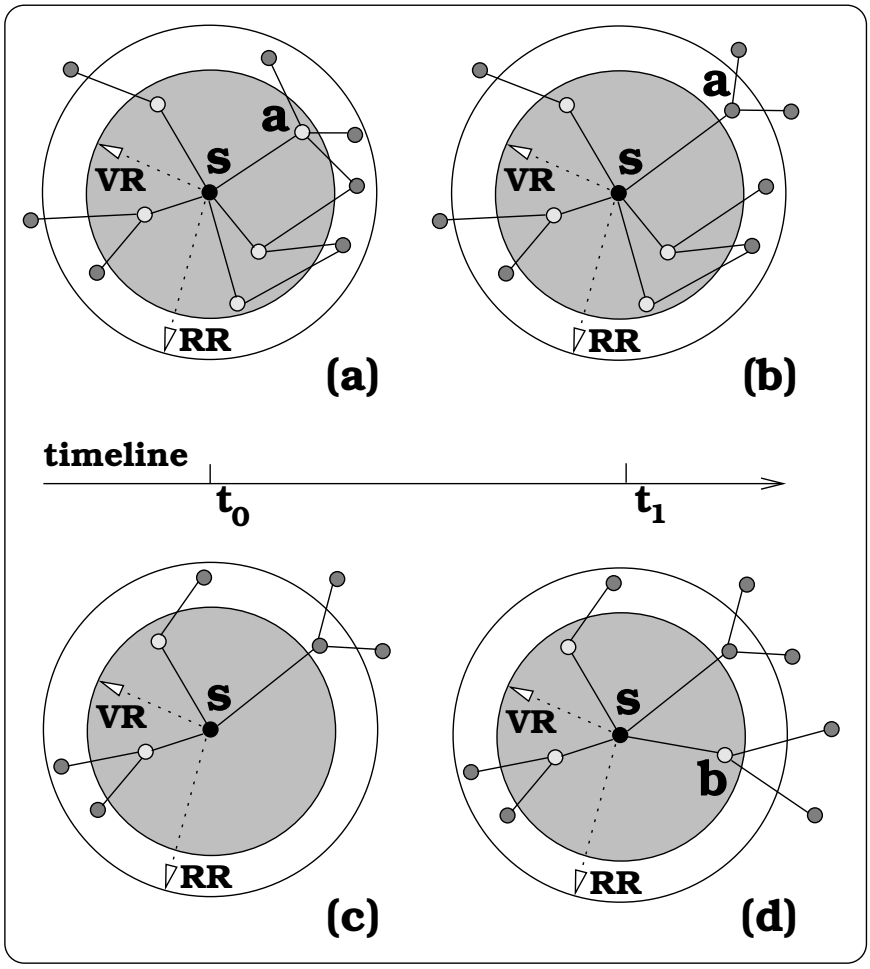



**Figure 4.12**: (a) When computing forwarders, only nodes within virtual radio range (VR) are considered one-hop neighbors. (b) Even though node $a$ moved out virtual range it is still within radio range (RR). (c,d) Node $b$ moves into virtual radio range of node $S$. In any previous HELLO message sent by $S$, node $b$ was not in the advertised $D^j_{\text{1-hop}}$ list. If after time $t_1$ node $S$ receives a broadcast for which it is not selected as a forwarder, and if no selected forwarder is covering node $b$ then $S$ will broadcast the packet anyway.

virtual radio range suffices. For the purpose of simulations, we assume that nodes exchange their location information using periodic HELLO messages.

Figure 4.12 (a,b) shows an example where a node, first within virtual range (i.e., node $a$), moves out of virtual range but is still within radio range. In this case, even though

node $a$ is no longer a one-hop neighbor for the purpose of forwarding computations, it is still reachable. Figure 4.12 (c,d) shows an example where a new node (i.e., node $b$) moves within virtual radio range. If node $S$ later on (i.e., after $t_1$) receives a broadcast packet for which it is not listed as a forwarder (i.e., $S \ni \mathcal{F}_{sender}$), node $S$ would still broadcast the packet in case node $b$ becomes a *one-hop dominating node*, and there is no forwarder in $\mathcal{F}_{sender}$ covering node $b$.

Figure 4.13 shows the pseudo-code for THEP. If a node is listed in the forwarder list (i.e., $\mathcal{F}_S$, available from the packet header), it means that the node must forward the packet. If that is not the case, recent modifications to the local topology may have changed the *one-hop dominating list*, and it may be different from the list used by the sender to compute the forwarder list. To check that, the node computes the *one-hop dominating list* (i.e., $D^i_{\text{1-hop}}$, advertised in periodic HELLO messages and used to compute the THEP forwarder list). If there is any node in $D^i_{\text{1-hop}}$ that is not covered by at least one forwarder in $\mathcal{F}_S$, then the node should relay the packet even though it has not been selected as a forwarder by the sender. In other words, if there is no forwarder in $\mathcal{F}_S$ covering any *one-hop dominating node*, then the broadcast might not reach the segment of the network connected to these *one-hop dominating nodes*.

### 4.2.1 Using THEP for Route Discovery

Similarly to AODV-THP, we have applied THEP as the basis for deciding which nodes should broadcast RREQ messages in AODV. The new protocol is named AODV-THEP, and it is implemented in *Qualnet* [1]. To address reliability, instead of requiring *at least one*

**Figure 4.13**: THEP

**Data**: $n_i$, $S$ (sender), $\mathcal{F}_S$ (sender's forwarder list), $D^k_{\text{1-hop}}$ for all $k \in N^i_1$
**Result**: $\mathcal{F}_i$, the forwarder list (if empty, do not broadcast)
**begin**
  $Broadcast = True$
  $\mathcal{F}_i \longleftarrow \emptyset$
  **if** $n_i \ni \mathcal{F}_S$ **then**
    $Broadcast = False$
    /* compute the *one-hop dominating list* using the most up-to-date neighborhood information */
    $D^i_{\text{1-hop}} \longleftarrow DP(N_i)$
    **for** $n_k \in D^i_{\text{1-hop}}$ **do**
      /* If there is any *one-hop dominating node* that is not covered by a node in $\mathcal{F}_S$ then broadcast */
      If $\nexists\, n_l \in \mathcal{F}_S \mid n_l \in N^k_1$ $Broadcast = True$
      Break
  **if** $Broadcast$ **then**
    $\mathcal{C} \longleftarrow N^i_1 - N^S_1$
    /* Select for each candidate *one-hop dominating nodes* other than one-hop neighbors and the node itself */
    **for** $n_k \in \mathcal{C}$ **do**
      $\mathcal{U}[k] \longleftarrow \emptyset$
      **for** $n_l \in D^k_{\text{1-hop}}$ **do**
        **if** $n_l \notin (N^i_1 + n_i)$ **then**
          $\mathcal{U}[k] \longleftarrow \mathcal{U}[k] + \{n_l\}$
    /* Exclude *one-hop dominating nodes* covered by another candidate in C */
    **for** $n_k \in \mathcal{C}$ **do**
      **for** $n_m \in \mathcal{U}[k]$ **do**
        **if** $\exists\, (n_l \neq n_k) \in \mathcal{C} \mid n_m \in N^l_1$ **then**
          $\mathcal{U}[k] \longleftarrow \mathcal{U}[k] - n_m$
          **if** $\mathcal{U}[k] == \emptyset$ **then**
            $\mathcal{C} \longleftarrow \mathcal{C} - n_k$
    /* For every node $n_k \in \mathcal{C}$, and for every $n_m \in \mathcal{U}[k]$, there is no other $n_l \in \mathcal{C}$ such that $n_m \in \mathcal{U}[l]$; therefore, all nodes in $\mathcal{C}$ are forwarders. */
    $\mathcal{F}_i \longleftarrow C$
  **return** $\mathcal{F}_i$
**end**

dominating node per two-hop neighbor (when computing the *one-hop dominating list*), every two-hop neighbor is covered by *at least two* one-hop nodes (if possible). This way, it increases the chance that a two-hop node receives a RREQ.

Routes to one-hop neighbors are kept as in standard AODV (i.e., nodes within physical radio range). Upon receiving a HELLO message, nodes update the route to the node sending the packet. The neighbor list advertised in the HELLO message contains only the neighbors within virtual radio range, and the $D_{\text{1-hop}}$ list is also computed using the virtual neighbor list.

Nodes relaying a RREQ packet, first compute the THEP forwarder list, update the RREQ header, and only then broadcast the packet. As in standard AODV, eventually the RREQ reaches a node with a valid route to the destination, or the destination itself (considering the network is connected). Because fewer nodes relay the same RREQ packet, we expect less contention and fewer collision of packets, as well as a smaller end-to-end delay, because the RREQ message propagates faster.

**Simulation Results**

The network is composed of $50$ nodes spread over an area of $1500m\ x\ 300m$. The radio model used is a $2Mbps$ IEEE $802.11$ device with a nominal transmission range of $250m$. Transmissions are omni-directional, and receptions are directional (this increases spatial reuse [65]). Traffic sources are continuous bit rate (CBR). Only $512$-bytes data packets are used. The source-destination pairs are chosen randomly among the nodes in the network. Flows last for $30s$ in average following an exponential distribution. Source nodes keep active flows

during the entire simulation time (new destinations are randomly selected as needed). During the simulation time, an average of $580$ flows are initiated, and at any given time there are $30$ active flows. Nodes begin transmitting at $50s$ plus an offset uniformly chosen over a $5s$ period to avoid synchronization in their initial transmissions. The simulation time is set to $600$ seconds, and identical mobility and traffic scenarios are used for all protocols. Initially nodes are placed uniformly over a grid. Nodes move according to the random way-point model with velocities between $1$ and $20m/s$. Six pause times are tested: $0s$ (always moving), $50s$, $100s$, $300s$, $400s$, and $600s$.

Experiments are repeated for $10$ trials with different random number seeds, traffic endpoints, and topologies. The topology and traffic pattern are fixed using off-line generated mobility and packet generation scripts. This means that all protocols are compared having identical node mobility and traffic demands. Each data point represents the average of the $10$ trials.

To evaluate the impact of the two enhancements, we run simulations for different virtual radio ranges. The following list summarizes all variants under consideration: AODV-THEP 1.0R, with virtual range and radio range the same (this way we can see the impact of the second enhancement alone); AODV-THEP 0.85R, with virtual range set to $85\%$ of the radio range; AODV-THEP 0.75R, with virtual range set to $75\%$ of the radio range; AODV-THP, AODV with standard THP; and AODV with and without HELLO messages.

Figure 4.14 presents the packet delivery ratio results. As expected, AODV-THP does not perform very well in scenarios with frequent topology changes. One of the main reasons is that it is more difficult to get an accurate view of the local topology when it changes

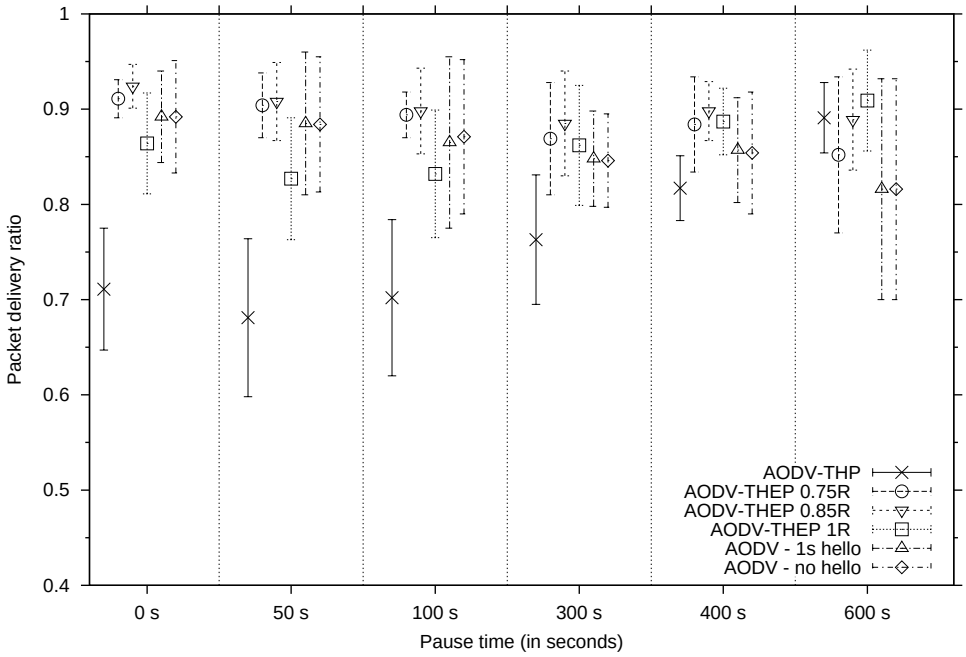



**Figure 4.14**: 50 nodes, 30 flows: packet delivery ratio

frequently. For static networks, AODV-THP delivers around $10\%$ more packets compared to AODV. AODV-THEP 1.0R shows the improvement due to the second enhancement by itself. It shows that it helps to compare any recent changes to the local topology to check if the sender is using any stale information (i.e., the last advertised *one-hop dominating list* may not include some new *one-hop dominating node*) when computing the list of forwarders (i.e., $\mathcal{F}$). AODV-THEP 1.0R starts performing better than AODV as mobility decreases (i.e., from $300s$ pause time and on), and it has the best results for static networks. Even though the topology actually does not change, because of transient link failures, and increased contention, the second enhancement helps to cope with transient changes to the local topology.

When mobility is present, the two proposed enhancements to THP operating to-

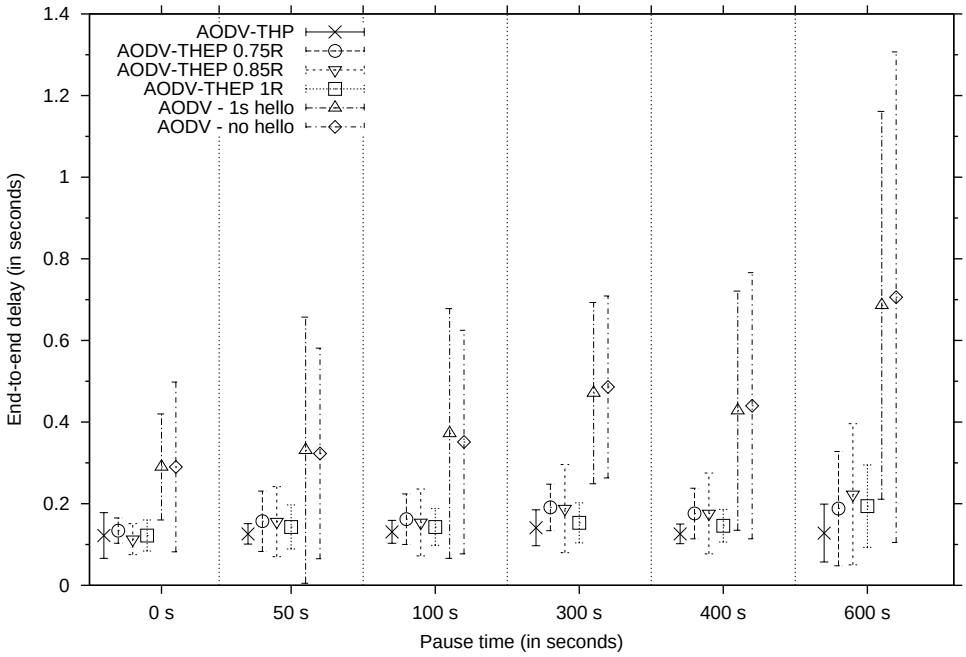



**Figure 4.15**: 50 nodes, 30 flows: average end-to-end delay

gether improve the performance of AODV in all circumstances. The results show that a VR of $0.85RR$ is better than $0.75RR$ for the scenarios under consideration. This means that a buffer zone of $0.15RR$ is enough to reach nodes moving out of the virtual range, and that it is better to keep more nodes within the VR when computing the forwarder list.

Figure 4.15 presents the average end-to-end delay results. Because around $580$ flows are initiated during the simulation time, we observe that the large number of redundant RREQ transmissions affect the end-to-end delay in AODV. AODV incurs twice to three times as much delay than any other variant. So, it pays off pruning redundant broadcast transmissions, because it reduces contention. The two enhancements proved to be effective in reducing delay, while sustaining a high delivery ratio. The extra control overhead introduced by periodic

HELLO messages in AODV does not have much impact on the end-to-end delay, because most of the routing load comes from RREQ transmissions. Together with the previous results for the delivery ratio, we can see that the reduction of redundant broadcast transmissions translate in a faster response to the route discovery process, which results in more packets being delivered at a smaller signaling cost.

Figure 4.16 shows the normalized routing overhead results. All the THEP variants present a much smaller overhead than AODV, because of the reduction on the number of redundant broadcast transmissions. As for the impact of the HELLO messages, we observe slightly more control overhead in AODV when HELLO messages are present. For static networks, AODV-THP presents the most cost-effective performance; its delivery ratio is the second, and it has the smallest end-to-end delay and control overhead. On the other hand, AODV-THEP shows better performance than the other protocols in high mobility scenarios.

Figure 4.17 presents the results for the number of collisions of packets. AODV with and without HELLOs presents similar results, showing that the introduction of HELLO messages is not responsible for increasing the number of collision of packets. On the contrary, the redundancy of RREQ transmissions is the cause for more contention and collisions. For static networks, AODV-THP presents the best overall performance with the only exception of a slightly smaller delivery ratio than AODV-THEP 1.0R. With mobility, even though the new enhancements incur slightly more packet collisions, they do improve the overall performance of the network by delivering more packets, with smaller delays, and less control overhead. Because there is a clear trade-off between efficiency and reliability, the two enhancements increase the reliability at the cost of increasing the number of redundant broadcast transmissions

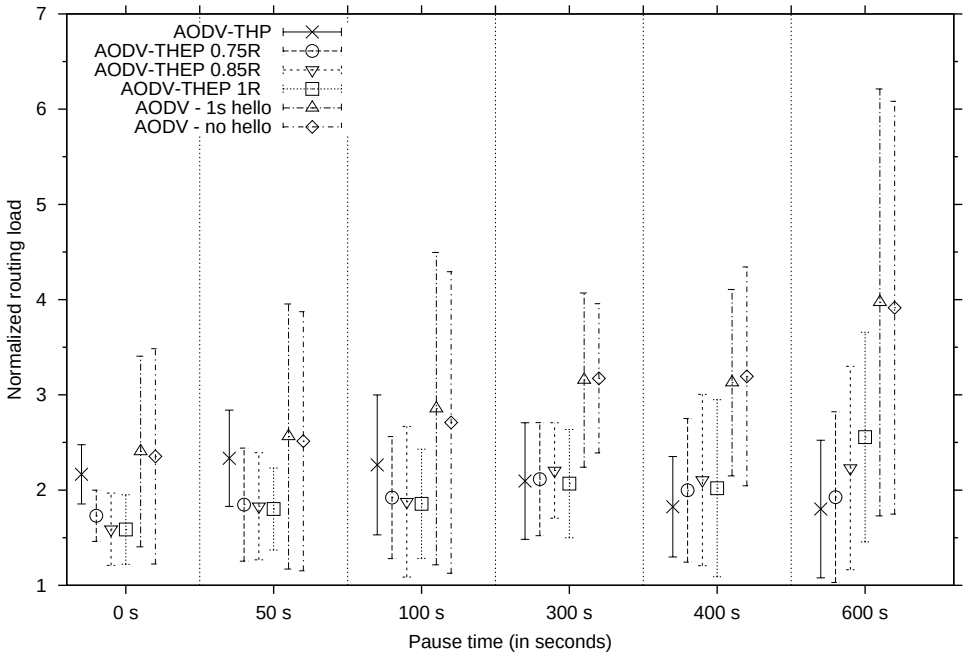



**Figure 4.16**: 50 nodes, 30 flows: normalized routing overhead

with respect to THP.

## 4.3 Conclusions

We presented THP, a localized algorithm for computing *two-hop connected dominating sets* (TCDS). In a TCDS, all nodes in the network are at most two-hops distant from some dominating node. We showed how THP can be applied to the route discovery process of on-demand routing protocols. The main contributions of THP are that *(a)* THP is the first heuristic to take into account three-hop information in the selection of relay nodes for the broadcasting of packets, while incurring signaling overhead that is much the same as that of heuristics based on two-hop information, and *(b)* THP reduces the number of redun-

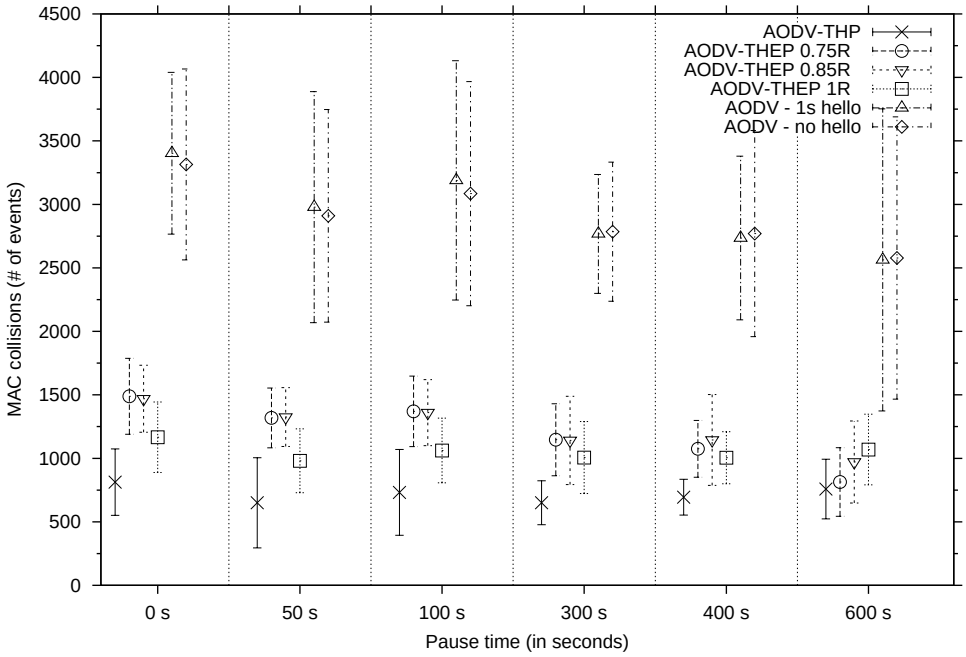



**Figure 4.17**: 50 nodes, 30 flows: number of MAC collisions

dant broadcast transmission. We show through extensive simulations that THP outperforms the best-performing self-pruning and neighbor-designated algorithms known when a TCDS is preferred over a CDS.

To improve the route discovery process of on demand routing protocols, THP is implemented in AODV (the new variant is named AODV-THP) as the mechanism for disseminating RREQ messages. The first simulation results show that THP improves, in all aspects, the performance of AODV in low mobility scenarios. We also show how to increase the reliability of THP (i.e., AODV-THP *2 cover*) by using *double coverage* instead of *single coverage* when computing the *one-hop dominating list*.

To address the lack of reliability in the presence of high mobility, we presented the

*Three-hop Horizon Enhanced Pruning* (THEP). First, a *virtual radio range* (VR), shorter than the physical *radio range* (RR), is used for gathering information about the two-hop neighborhood. Instead of using two different transmission powers, which can incur additional interference, we use a single transmission power while still managing to have a *buffer zone* in which neighbors can move without compromising network connectivity. Second, upon receiving a broadcast packet, the forwarder list in the packet header is analyzed together with the current information about the local neighborhood. This is done to find inconsistencies between the most up-to-date *one-hop dominating list* and the one used by the sender to compute the sender's forwarder list. Changes in the local topology may have impacted the *one-hop dominating list*. If that is the case, a node may decide to relay a broadcast packet even though it was not selected as a forwarder by the sender.

Extensive simulation results show that AODV-THEP attains better performance than AODV for all mobility scenarios in terms of delivery ratio, control overhead, packet collision, and end-to-end delay.

# Chapter 5

# Bounded-Distance Multi-Clusterhead Formation in MANETs

In mobile ad hoc networks (MANETs), hierarchical architectures can be used to prolong the network's lifetime [6, 10, 73], attain load balancing [8], and increase network scalability [25, 34]. *Clustering* is the problem of building a hierarchy among nodes [12]. The substructures that are collapsed in higher levels are called *clusters*. In each cluster, at least one node may represent the cluster, and this node is usually called a *cluster-head*. The network can then be abstracted such that any cluster-head connects to another cluster-head whenever there is at least one node in each cluster directly connected to each other.

Clustering usually entails the computation of a *dominating set* (DS) of the network. The domination problem seeks to determine a minimum number of nodes $D$ (called *dominating nodes* or *cluster-heads*), such that any node $i$ not in $D$ is adjacent to at least one node in $D$. The computation of a DS of minimum cardinality for arbitrary graphs is known to be

NP-complete [19].

A variety of conditions may be imposed on the dominating set [24]. The $(k, r)$-DS problem [31] has been defined as the problem of selecting a minimum cardinality vertex set $D$ of a graph $G = (V, E)$, such that every vertex $u$ not in $D$ is at a distance smaller than or equal to $r$ (*distance domination*) from at least $k$ (*multiple domination*) vertices in $D$. The problem of computing a $(k, r)$-DS of minimum cardinality for arbitrary graphs is also NP-complete [31].

When selecting dominating nodes, redundancy is achieved by choosing a value for the parameter $k$ greater than one. At the same time, the distance parameter $r$ allows increasing local availability by reducing the distance to the dominating nodes. Depending on the requirements, problems that require the computation of a DS can be solved by setting the two dominating parameters appropriately.

The $(k, r)$-DS could be applied to solve a large variety of problems, of which we mention three below.

*Hierarchical Routing:* By grouping clusters into super-clusters, and so on, an $m$-level hierarchical clustering [32] structure can be built. Some approaches [62] target wired networks assuming that the predefined hierarchical address of each node reflects its position within the hierarchy. During the early days of *packet radio networks* (PRNET's), hierarchical routing had been considered for reducing the routing cost and improve the performance of the network [49]. In MANETs, group mobility is usually assumed when deriving hierarchical clusters [43, 44]. However, none of the existing solutions address redundancy within the hierarchical structures. Instead of having just one cluster-head representing a group of nodes, hierarchies could account for node failures by deploying multiple cluster-heads within each

sub-structure (i.e., domain). Nodes within the same domain would have alternate access points when accessing nodes outside their own domain, and adjacent domains could be connected among each other through alternate paths. Furthermore, load balancing could be explored by routing packets via alternate paths connecting any pair of nodes.

*Core Placement in Shared-Tree Multicasting:* Instead of deploying just one core per multicast group (e.g., like in *Core Based Trees* [5]), multiple cores can be selected within regions of variable radius in the network. That is, multicast members have the choice of joining multiple cores within a maximum distance. More than one core provides a certain degree of fault-tolerance, and a maximum distance to the cores can reduce the average end-to-end delay.

*Demand-Driven Applications in Sensor Networks* [2]: Multiple sinks can be distributed in a sensor network to provide some degree of fault-tolerance, and a maximum distance from nodes to sinks could support a bounded report delay. Sinks could also be organized in a multi-level hierarchy, where sinks aggregate data collected from their domain, and transmit it to a higher level sink.

We propose the first centralized and distributed solutions to the $(k, r)$-DS problem for arbitrary network topologies. The centralized solution provides an approximation to the optimal solution, and is used as a lower bound when evaluating the performance of the distributed solution. The distributed solution is applicable to ad hoc networks, given that it relies on information limited to the neighborhoods of nodes.

## 5.1 (k,r)-Dominating Sets: Centralized Solution in Arbitrary Graphs

The centralized solution presented in this Section, KR, requires that the entire network topology be known. Hence, if the solution were used in a network, the network topology would have to be broadcast, and the dominating set computed by each node in the network would be the same for any node.

### 5.1.1 Description

Any node $i$ is said to be $(k,r)-$dominated (or simply dominated) if node $i$ has at least $k$ neighbors within distance $r$ in $D$ (for notation refer to the List of Notations).

For the computation of a DS with parameters $k$ and $r$, the $Domin$ value of node $i$, $i.Domin$, is $k$ minus the number of nodes in $D$ within distance $r$ from $i$ if node $i$ is not covered (or dominated). Once node $i$ has been dominated, or node $i$ is selected as dominating, its $Domin$ is set equal to zero, and no longer changes its value. The $total$ value of node $i$, $i.total$, is given by $i.total = \sum_{k \in \aleph_{r,i}} k.Domin$.

A natural greedy solution to the $(k,r)$-DS problem would be to repeatedly select as *dominating* the node that covers the most number of uncovered nodes (i.e., nodes not yet $(k,r)$-dominated), until all nodes are $(k,r)$-dominated. However, KR applies a different greedy approach, and repeatedly selects as *dominating* the node with the current largest *total* value. That is, KR selects as *dominating* the node that covers the most number of nodes with fewer dominating nodes (it could be the node with the most number of nodes not yet covered), which is quantified by the *total* parameter of any node. This way, any selected node

$i$ potentially affects the *total* value of any node within distance $2r$ from node $i$. On the other hand, in the natural greedy approach, any selected node $i$ only affects the coverage of nodes when some node in node $i$'s $r$-hop neighborhood gets $(k, r)$-covered. That is, any selected node $i$ reduces by one the $Domin$ value of any node $n$ not yet covered in node $i$'s $r$-hop neighborhood, but it does not reduce by more than one unit node $n$'s coverage if no node gets $(k, r)$-covered other than node $i$ itself. For *multiple domination* one (i.e., $k = 1$), KR is equivalent to the natural greedy approach (i.e., the first time any node $i$ gets covered, its $Domin$ value becomes zero).

Figure 5.1 presents a pseudo-code for KR. Initially, $Domin$ is set equal to $k$ for all nodes, and $total$ depends on the number of nodes in the $r$-hop neighborhood of each node. Nodes are inserted in a *Heap* structure to make the selection of the node with the largest *total* value easier. At the beginning, all nodes are in the *Heap*, and while there are nodes in the *Heap*, there are nodes yet to be covered. The node with the largest $total$ is selected as the next dominating node. Once node $s$ is selected, all nodes in its $r$-hop neighborhood that are not yet covered (i.e., $Domin > 0$), must have their $Domin$ value recalculated to reflect the selection of node $s$. Nodes that become covered (i.e., $Domin = 0$) are removed from the *Heap*. After this update, nodes in node's $s$ $\mathbf{2r}$-hop neighborhood remaining in the *Heap* must have their $total$ value recalculated. Because nodes within distance $r$ from node $s$ may have their $Domin$ value changed, it implies that nodes within distance $2r$ from $s$ may have their $total$ value affected as well. After these updates, the $Heap$ must be sorted, so that the node with the largest $total$ can be selected next (ties are broken choosing the node with lowest ID). This process repeats, until the $Heap$ is empty.

**Figure 5.1**: KR

```
Data: G = (V, E), k, r
Result: D, the (k, r)-DS
begin
    D = ∅
    /* Initialize parameters Domin and total                    */
    foreach i ∈ V do
        i.total = k · |ℜ_{r,i}|
        i.Domin = k
        /* Insert i in the HEAP (sorted according to
           the total parameter).  Ties are broken
           choosing the node with the lowest ID.                */
        Insert(HEAP, i)
    /* Select dominating nodes                                  */
    while HEAP ≠ ∅ do
        /* Node with largest total is the next
           dominating node                                      */
        s = First(HEAP)
        s.Domin = 0
        D = D ∪ {s}
        /* Update Domin value for each covered node             */
        foreach i ∈ ℜ_{r,s} | i ∈ HEAP do
            − − i.Domin
            /* If a node becomes dominated remove it from
               heap                                             */
            if i.Domin == 0 then
                Remove(HEAP, i)
        /* Update total of nodes affected by the
           current selection                                    */
        foreach i ∈ HEAP | i ∈ ℜ_{2r,s} do
            i.total = 0
            foreach j ∈ ℜ_{r,i} do
                i.total = i.total + j.Domin
        Sort(HEAP)
end
```

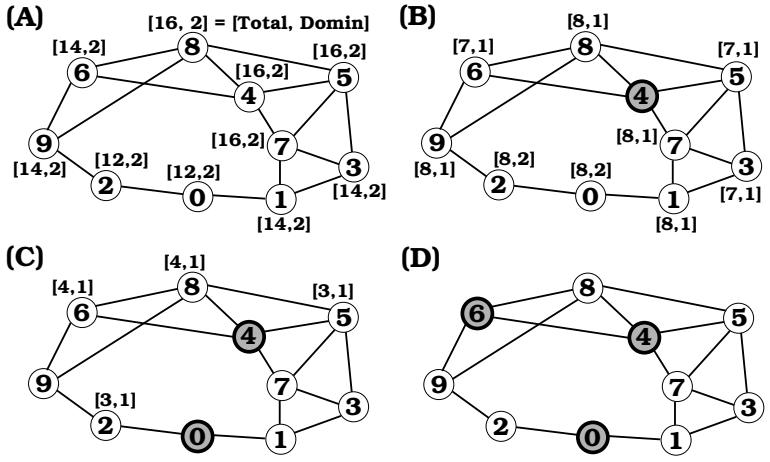



**Figure 5.2**: Computing a $(2,2)-$DS of the network with KR.

### 5.1.2 Example

Figure 5.2 depicts an example of KR computing a $(2,2)$-DS of the network. Initially nodes have their $Domin$ parameter set to 2, and the $total$ parameter is computed depending on each node's two-hop neighborhood (Figure 5.2 (A)). Nodes $\{4,5,7,8\}$ have the same largest $total$, but recall that in KR ties are broken lexicographically; hence, node $4$ is selected as dominating. Figure 5.2 (B) shows the network reflecting the selection of node $4$, and once again there are several nodes with the same largest $total$ (i.e., nodes $\{0,1,2,7,8,9\}$). In this case, node $0$ is selected. After its selection (Figure 5.2 (C)), nodes $\{6,8\}$ have the same largest total. Node $6$ is then selected, which makes all nodes $(2,2)$-Covered (Figure 5.2 (D)). If the natural greedy approach were applied to the same network, then the order of nodes selected would be $\{4,5,0,2\}$, increasing the cardinality of the DS by one node.

### 5.1.3 Analysis of KR

In this section we show that KR computes a $(k, r)$-DS of any arbitrary network in polynomial time, yielding a dominating set of size at most $k \cdot \ln \Delta$ times the optimal.

**Correctness and time complexity**

Lets consider an arbitrary graph $G = (V, E)$ of order $n = |V|$, represented using adjacency-lists. The initialization takes $O(n \cdot \log n)$ time, because each node in the graph needs to have its *Domin* and *total* parameters set, after which the node is inserted in the $HEAP$. Following the initialization, the selection of dominating nodes takes place. Dominating nodes are selected from the $HEAP$. The *while* loop is executed $O(n)$ times. The first *for* loop is executed $O(\Delta)$ times, and removing a node from the $HEAP$ takes $O(\log n)$, what gives a $O(\Delta \cdot \log n)$ time for this internal loop. The second *for* loop is also executed $O(\Delta)$ times, and its internal *for* loop is executed $O(\Delta)$ times, what gives a $O(\Delta^2)$ time for the second *for* loop. Given that $\Delta \leq n$ (i.e., $\Delta$ increases as $r$ approaches the network diameter, and is at most $n$ when $r = diameter$), KR runs in $O(n^3)$ time.

**Theorem 3.** *KR correctly computes a $(k, r)$-dominating set of any connected graph $G = (V, N)$.*

*Proof.* Initially all nodes are in the *Heap*. While there are nodes in the *Heap*, the node at the top (i.e., with the largest *total* value) is selected as dominating. A node is also removed from the *Heap* when its $Domin$ value is equal to zero (i.e., the node is $(k, r)$-dominated). Hence, when the *Heap* is empty, any node in the graph is either $(k, r)$-dominated or dominating. □

**Approximation ratio**

**Theorem 4.** *KR with parameters $k$ and $r$, yields a dominating set of size at most $\mathbf{k} \cdot \ln \Delta \cdot |\mathbf{OPT_{DS}}|$, where $OPT_{DS}$ is an optimal $(k, r)$-dominating set in the graph.*

*Proof.* Let $OPT_{DS}$ be the set of vertices in an optimal $(k, r)$-dominating set. The sets of vertices of $G$ dominated by vertex $i \in OPT_{DS}$ is called $S_i$ (assuming that $i$ also belongs to $S_i$). If a vertex is dominated by more than $k$ dominating nodes, we arbitrarily put it in $k$ of such sets. The proof is based on amortized analysis. Each time a dominating node $y$ is selected, the operation has cost 1. This cost is equally distributed among all newly covered vertices in $\mathfrak{N}_{r,y}$. We then prove that the total charge on the vertices belonging to a set $S_i$ (for any $i$) is at most $k \cdot \ln \Delta$. Since there are $|OPT_{DS}|$ sets in the optimal solution, the theorem follows.

Assume that $N$ vertices are covered when a node $s$ is selected as dominating. We charge each such *newly* marked vertex $\frac{1}{N}$, and a vertex can be marked at most $k$ times. That is, while a vertex has been marked less than $k$ times, it is considered as *newly* marked.

We now prove the upper bound on the total charges to vertices belonging to a single set $S_i$. At each step, some vertices may get marked. The number of unmarked vertices in $S_i$ is initially $u_0$, and finally drops to 0. In step $j$, the number of vertices marked in $S_i$ is $u_j - u_{j+1}$. For simplicity, it is assumed that some vertices of $S_i$ get marked at each step, decreasing the number of unmarked vertices.

In the worst case, no two nodes in $S_i$ are covered together. In this case, during step $j$ each node in $S_i$ can be charged at most $\frac{1}{u_j}$. Otherwise, node $i$ could be selected, because it would have covered at least $u_j$ nodes. Let $u_m = 0$ (i.e., at step $m$, all nodes in $S_i$ have already

been covered). Adding up all the charges (assuming nodes are charged at most $k$ times) we get:

$$
\begin{aligned}
k \cdot \sum_{j=0}^{m-1} \frac{u_j - u_{j+1}}{u_j} &= k \cdot \sum_{j=1}^{m} \frac{1}{u_{j-1}}(u_{j-1} - u_j) && (5.1)\\
&\leq k \cdot \sum_{j=1}^{m} H(u_{j-1}) - H(u_j) && (5.2)\\
&= k \cdot (H(u_0) - H(u_m)) && (5.3)\\
&= \mathbf{k} \cdot \ln \Delta && (5.4)
\end{aligned}
$$

(Where $H(d) = \sum_{i=1}^{d} \frac{1}{i} = \ln d + O(1)$, $H(b) - H(a) \geq \frac{b-a}{b}$, $H(0) = 0$, and assuming the fact that $u_0 \leq \Delta$)

□

## 5.2 Distributed Clustering Using (k,r)-Dominating Sets

As discussed previously, the *distance* and the *multiple* domination parameters can be used to define the degree of redundancy for bounded-distance clusters. That is, the two dominating parameters define the maximum distance from nodes to their cluster-heads, and the minimum number of cluster-heads per node, respectively.

We propose DKR, which is a distributed algorithm for clustering using $(k, r)$-DS. DKR is well suited for both synchronous and asynchronous networks. In the synchronous network model, nodes exchange messages in synchronous rounds. In the asynchronous network model, nodes take steps at arbitrary times. Even though there are no rounds in the asynchronous model, it is possible to simulate rounds [17]. In order to do that, a node tags the

message with its round number $x$. The recipient waits to receive round $x$ messages from all its neighbors before transitioning to the next round.

### 5.2.1 Summary Description

We assume that nodes have unique identifiers (IDs), and that nodes know who their neighbors are. The latter can be implemented by means of a neighbor protocol with which nodes exchange hello messages [39], as part of the MAC protocol, or using periodic *HELLO* messages as part of the protocol itself.

Associated with any node $i$ in the network, there is a *process* that consists of the following components: A set of *states*, which is used for describing the current state of node $i$. A message-generation function that specifies any messages that node $i$ should send and to whom it should send them, depending on the current state of the system. Optionally, a list of *events*, each of them scheduled to happen at a specific time. A state-transition function specifying the new state to which node $i$ should transition for each possible state and messages received.

The *status* of a node reflects its role during the *clustering* process. Initially, there is no established hierarchy among nodes, and the nodes assume an *unknown* status. As the nodes organize themselves, their status change to reflect their role in a cluster, which can be one of the following:

- *Dominating*, the node is a *cluster-head*.
- *Pending Dominating*, the node may become a *cluster-head*.
- *Dominated*, the node has *at least* $k$ cluster-heads within distance $r$.

- *Gateway*, in addition to being *dominated*, the node connects other nodes to their cluster-heads.

A **round** of messages is defined as the successful transmission of a message $\mathbf{m}$ by any node $n$ to all its one-hop neighbors. If rounds are numbered, a round $x$ is deemed complete only after all nodes have sent the messages for round $x$. DKR has two phases:

- *Phase One* (Election Phase): Each node elects $k$ nodes with smaller IDs (possibly including the node itself) within distance $r$. Elected nodes are just *candidates to be cluster-heads*. Because each node has its own set of $k$ elected nodes within distance $r$, the sets of elected nodes *dominate* all non-elected nodes in the network.

- *Phase Two*: During this phase *cluster-heads* are assigned, and nodes are affiliated to their cluster-heads.

Clearly, there must be at least $k$ nodes in every node's $r$-hop neighborhood for the required *multiple domination* to be satisfied. In the subsequent description of DKR, we assume that multiple domination can be satisfied at each node.

It is possible that not all nodes elected during *Phase One* become *cluster-heads*, because some redundant candidates are identified, and pruned. The rationale for choosing node IDs over node degree for the election process is that elections based on node degree can result on high turnover of dominating nodes when the topology changes, because the degree of a node is much more likely to change than the node ID relative to its neighborhood [21].

**Phase One**

This phase takes $r$ rounds to complete in a static topology. A pseudo-code for *Phase One* is presented in Figure 5.3. For asynchronous networks, rounds are simulated as described previously. At the beginning of a new round, a node advertises its list of $K \leq k$ smaller ID nodes. After a number of rounds ($r$ rounds in a static topology), a node $i$ in the network learns the set of $k$ nodes with smaller IDs (possibly including the node itself) within distance $r$ from it. We denote such a set by $D_i'$.

An elected node can elect itself or be elected by other nodes. A node that elects itself is called *properly-elected* if the node is not elected by any other node, and is called *self-elected* if the node is elected by at least one other node. A node that does not elect itself and is elected by other nodes that are not elected is called *neighbor-elected*. After the election, any node $i$ in the network changes its status as follows: If node $i$ is properly-elected or self-elected, node $i$ changes status to pending dominating. Otherwise, node $i$ has status dominated.

Note that a *properly-elected* node must become dominating, because there are at most $k-1$ other elected nodes in node $i$'s $r$-hop neighborhood. Because identifying properly-elected nodes would incur extra overhead, they are implicitly notified of their dominating status after not hearing from enough dominating nodes within a given period of time.

A *neighbor-elected* node $i$ is elected by at least one node, call it $n$, which is not elected and for which node $i$ is strictly required. That is, there is no self-elected node in node $n$'s $r$-hop neighborhood that could possibly replace node $i$; otherwise, node $n$ would have elected that self-elected node. Even though in some cases a properly-elected node could replace node $i$, initially DKR chooses to select all neighbor-elected nodes as cluster-heads.

**Figure 5.3**: DKR: *Phase One* (Election Phase)

**states consists of:**

| | |
|---|---|
| $i$ | the node ID |
| $k, r$ | the dominating set parameters |
| $D_i^{'}$ | set of tuples, with each tuple specifying a node ID, the node advertising it (i.e., next-hop toward the node), and the distance to the node. **Initially** contains only the tuple $\{i, i, 0\}$ |
| $A$ | set formed from set $D_i^{'}$, with each tuple specifying a node ID and its known distance (i.e., first and third parameter of each tuple from set $D_i^{'}$) |
| $H$ | set of all nodes known during the election process |
| *status* | {*unknown, pending_dominating, dominating, dominated*}, **initially** *unknown* |
| *rounds* | an integer, initially 0 |
| *NA* | set of nodes requiring *Neighborhood Advertisement* (**initially** $NA = \emptyset$). |

**messages:**

**if** $rounds < r$ **then**
- `/* send current list of known` $K \leq k$ `known smallest IDs */`
- send $A$ to all neighbors

**transitions:**

$rounds = rounds + 1$
$M = \{$set of tuples received from neighbors$\}$
**foreach** $t = \langle domin, dist \rangle \in M$ **do**
- **if** $\exists \langle a, b, c \rangle \in D_i^{'} \mid a == domin,\ (dist + 1) < c$ **then**
  - `/* found shorter path to some node in set` $D_i^{'}$ `*/`
  - $b =$ *node sending tuple* $t$
  - $c = dist + 1$
  - update correspondent entry in set $H$
  - $M = M \setminus \{t\}$

`/*` $H$`: all nodes known during the election process */`
$H = H \cup M$
$P = D_i^{'} \cup M$
$D_i^{'} = \{min(k, |P|)$ smaller IDs in $P\}$
update $A$ with new set $D_i^{'}$
**if** $rounds == r$ **then**
- `/* Election is over! */`
- **if** $i \in D_i^{'}$ *OR* $|D_i^{'}| < k$ **then**
  - **if** $|D_i^{'}| < k$ **then**
    - `/* Not satisfiable! */`
    - $status =$ *dominating*
    - `/* Must send an NA message during` *`phase two`*`,`
      `advertising I am a` *`cluster-head`* `*/`
    - $NA = \{i\}$
  - **else**
    - $status =$ *pending_dominating*
  - $D_i^{'} = \emptyset$
- **else**
  - $status =$ *dominated*
- `/* go to` *`Phase Two`* `*/`
- DKR: *Phase Two*

**Figure 5.4**: DKR *Phase Two*: **states**, **messages**, and **events**

**states consists of (in addition to states from *phase one*):**

- $N$ set of nodes requiring Notification (**initially** $N = \emptyset$)
- $LA$ **initially** *True*
- $J$ set of nodes requiring a *Join* message (**initially** $NA = \emptyset$)

**messages:**

**if** *LA == True* **then**
- `/* only` *dominated* `nodes send Local Advertisement */`
- **if** *status == dominated* **then**
  - Broadcast $\langle$LocalAdvertisement, $D_i^{'}\rangle$
- **else**
  - `/* Schedule event for checking dependencies */`
  - ScheduleEvent(CheckStatus, *Wait Period*)
- LA = False;

**if** $N \neq \emptyset$ **then**
- **foreach** $\langle target, next\_hop\rangle \in N$ **do**
  - Unicast to $next\_hop$ $\langle$Notification, $target\rangle$;
- N $= \emptyset$

**if** $NA \neq \emptyset$ **then**
- `/* Initiating or relaying a neighborhood advertisement */`
- **foreach** $n \in NA$ **do**
  - *Broadcast* $\langle$NeighborhoodAdvertisement, $n\rangle$
- $NA = \emptyset$

**if** $J \neq \emptyset$ **then**
- **foreach** $n \in J$ **do**
  - $l =$ get next hop to $n$ from $H$
  - Unicast to $l$ $\langle$Join, $n\rangle$;
- J $= \emptyset$

**events:**

**case** *CheckStatus*
- $L = \{$validated nodes in set $D_i^{'}\}$
- **if** *status == pending_dominating* **then**
  - **if** $|L| < k$ **then**
    - `/* not enough dominating nodes within distance` $r$ `*/`
    - *status = dominating*
  - **else**
    - **if** *any Notification relayed by this node* **then**
      - *status = gateway*
    - **else**
      - *status = dominated*
- **if** *status* $\neq$ *dominating* **then**
  - `/* send a` *Join* `message to all my dominating nodes */`
  - **if** $|L| < k$ **then**
    - $F = D_i^{'} \cap L$
    - `/* Chooses` $k - |L|$ `nearest ones! */`
    - $L^{'} = \{k - |L|$ closest nodes in $F\}$
    - $J = L \cup L^{'}$
    - Validate entries in $L^{'}$
  - **else**
    - $J = L$

**Phase Two**

During *phase two*, some or all nodes elected during *phase one* become cluster-heads. In addition, the rest of the nodes are affiliated to their cluster-heads. A pseudo-code for phase two is presented in Figures 5.4 and 5.5.

The messages used during this phase are:

- *Local Advertisement* (LA): A message having the list of nodes elected by the sender, and the respective next-hop to each one of the elected nodes.
- *Neighborhood Advertisement* (NA): A message advertising a cluster-head.
- *Notification*: A message sent to notify a node that must become cluster-head.
- *Join*: A message sent to notify, or to connect to a given cluster-head.

Because neighbor-elected nodes are not aware of their election, a notification mechanism is needed to notify them. Depending on the coverage provided by neighbor-elected nodes, some self-elected nodes may be ruled out as cluster-heads.

At the beginning of *phase two*, dominated nodes send to their one-hop neighbors an LA message containing their elected nodes. Any dominated node $i$ proceeds as follows upon receiving an LA message:

- If node $i$ is listed in the advertisement, node $i$ changes its status to dominating, triggering an NA message announcing node $i$ as cluster-head to all its $r$-hop neighbors. This is accomplished by broadcasting the NA message using restricted *blind-flooding* with the *time-to-live* (TTL) field set equal to $r$.

- If node $i$ is not listed in the LA message but is listed as a next hop to any advertised node, then node $i$ changes its status to gateway.

- For any advertised node $n \in LA$ that is not among the nodes elected by node $i$ (i.e., $n \ni D_i^{'}$) it sends a *Notification* message to node $n$. Upon receiving the notification, if the notified node is not yet dominating, the node advertises itself via an NA message.

**Definition 1.** *For any node $i$, and for all $n \in D_i^{'}$, node $n$ is deemed **validated** only upon the reception of the respective NA message advertising node $n$; otherwise, node $n$ is not yet **validated**.*

Any neighbor-elected node $n$ eventually changes its status to dominating, by either receiving a *Notification* message originated at some node $k$ within distance $1 < d < r$, or by receiving an LA message from some one-hop neighbor. In any case, once node $n$ becomes dominating, it sends an NA message.

Because an NA message is sent only when a node changes its status to dominating, nodes receiving an NA message advertising node $n$ know that it is a cluster-head. When processing an NA message for node $n$, any node $i$ inserts an entry for node $n$ in $D_i^{'}$.

A *pending dominating* node $i$ does not become a *cluster-head* if ***(a)*** node $i$ has *at least* $k$ validated dominating nodes within distance $r$, and ***(b)*** every node $n$, which elected node $i$ during *phase one*, is also covered by a set of validated dominating nodes.

**Definition 2.** ***Wait Period** is the minimum time required for reaching an agreement in* phase two.

In the worst case, a *Notification* for node $n$ is initiated by some neighbor $i$ located

$r-1$ hops away from node $n$. The correspondent NA message initiated by node $n$ reaches the most distant neighbors (i.e., $r$ hops away from node $n$) after $r$ successfully transmissions of message NA. Therefore, **Wait Period** should be larger than the time required for $2r$ transmissions of a message. After a period of time equivalent to *Wait Period* any node $i$ in the network checks its coverage. If node $i$ is *pending dominating*, and it does not have enough validated entries in $D_i^{'}$, then node $i$ changes status to *dominating*, and sends an NA message. This means that node $i$ does not have enough information for ensuring its own coverage (i.e., node $i$ does not know if it has at least $k$ dominating nodes within distance $r$). Otherwise, any non-*dominating* node $i$ sends a *Join* message to $k$ nodes from $D_i^{'}$ (including the nodes already validated). If there are more than $k$ validated entries in $D_i^{'}$, node $i$ chooses the closest ones (ties are broken choosing the node with smaller ID).

Like a *Notification*, which also serve for assigning *gateways* while the message is being routed to its destination, *Join* messages also serve to notify any *pending dominating* node that is still required as cluster-head. That is, even though some *pending dominating* node $i$ finds itself covered, there might be some node $j$ that still needs node $i$ (i.e., without node $i$, node $j$ does not have the required number of dominating nodes). While a *Join* message is being routed to destination $n$, a node $i$ processing the message does not need to relay it if a *Notification*, or another *Join* message, had already been sent to node $n$. So, we assume that every node keeps track of recent *Notification* and *Join* messages sent by the node. After *Join* messages reach their destinations, all regular nodes are connected to their cluster-heads.

**Figure 5.5**: DKR *Phase Two*: **transitions**

```
transitions:
  M = {message(s) from neighbor(s)}
  foreach m ∈ M do
    Sender = node that sent message m
    switch m do
      case LocalAdvertisement
        /* local advertisements consist of tuples ⟨a, b, c⟩, where
           a=dominating node, b=next hop, and c=distance          */
        foreach ⟨a, b, c⟩ ∈ m do
          if a == i AND status≠ {dominating OR pending_dominating} then
            status=dominating
            NA = NA ∪ {i}
          else
            if status==dominated AND b == i then
              status=gateway
            /* If a is not one of my selections, and I am on the
               path to node a, send a notification to a          */
            if a ∋ D_i' AND b == i then
              n = get next hop to a from set H
              if n ≠ Sender then
                N = N ∪ {⟨a, n⟩}
      case Notification
        if target == i then
          /* I am being notified                                 */
          if status ≠dominating then
            status=dominating
            NA = NA ∪ {i}
            D_i' = ∅
        else
          if target ∋ D_i' then
            n = get next hop to target from set H
            if n ≠ Sender then
              if status == dominated then
                status=gateway
              if Notification not previously sent to target then
                N = N ∪ {⟨target, n⟩}
      case Join
        if target ≠ i then
          if Join OR Notification not yet sent to target then
            if status == dominated then
              status=gateway
            J = J ∪ {n}
      case NeighborhoodAdvertisement
        a = advertised node
        if status ≠ dominating then
          if a ∋ D_i' then
            D_i' = D_i' ∪ {a}
        if (distance to a) < r then
          /* relay advertisement                                 */
          NA = NA ∪ {a}
```

### 5.2.2 Examples

Consider the example presented in Figure 5.6, where nodes are computing a $(2,3)$-DS of the network. During *Phase One* nodes elect the two nodes with smaller IDs in their 3-hop neighborhood (Figure 5.6(A)). Nodes $10$ and $15$ are *self-elected*, and nodes $18$ and $20$ (both elected by node $90$) are *neighbor-elected*. *Self-elected* nodes assume status *pending dominating*. The remaining nodes assume status *dominated*, and send an LA message advertising their list of elected nodes. After receiving the LA message from neighbor $90$, node $18$ changes status to *dominating*, and sends an NA message that eventually reaches all nodes within distance $3$ from node $18$. Figure 5.6(B) show the status of nodes, and their corresponding *validated* dominating entries. Besides sending the NA message, node $18$ also sends a Notification to node $20$. Figure 5.6(C) presents the status of the network after the notification has reached node $20$. The notification makes node $20$ change its status to *dominating*, triggering an NA message that eventually reaches all neighbors within distance $3$ from node $20$. After all affected nodes process this NA message, we notice that all dominated nodes are satisfied, because each of them have $2$ validated entries in their $D^{'}$ lists. *Wait Period* should be set appropriately, so that by the time the event *CheckStatus* happens all NA messages have already been delivered and processed. Figure 5.6(D) shows the status of the network after all dominated nodes have sent out their *Join* messages to their dominating nodes (assume that *Join* messages are grouped together whenever different dominating nodes are reachable through the same node). In this case, because nodes $50$ and $80$ have either relayed a *Notification*, or a *Join* message, they serve as *gateways* for other dominated nodes.

Figure 5.7 presents an example comparing DKR (*assuming maximum ID nodes are*

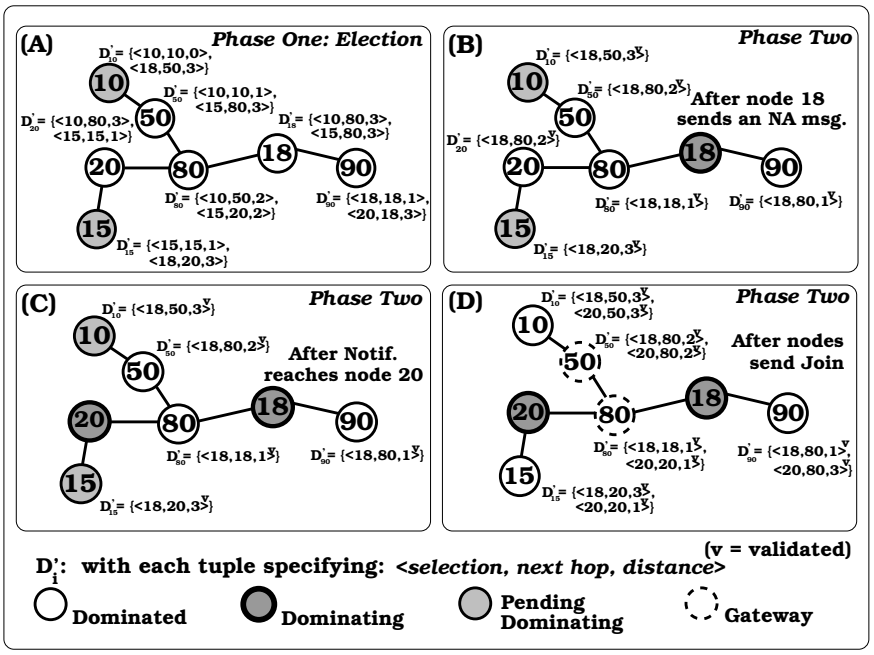



**Figure 5.6**: Computing a $(2, 3)$-DS of the network

*elected, rather than the default minimum ID*) to *Max-Min* [3], when computing a $(1, 3)$-DS of the network. This is the same network example presented in [3]. While *Max-Min* produces $4$ cluster-heads, DKR produces $3$. For DKR, node $100$ is a *self-elected* node. The other elected nodes (i.e., $65$, $73$, and $85$) are eventually notified, and announce themselves by sending NA messages. All nodes that have elected node $100$ (including node $100$ itself) receive at least one of these NA messages, validating the respective advertised node. This is an indication that the election process can be improved, producing smaller dominating sets. Simulation results presented latter corroborate this.

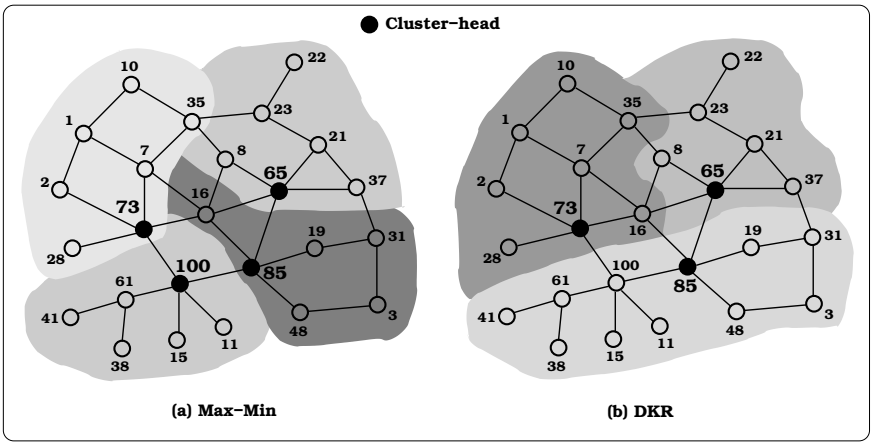



**Figure 5.7**: Computing a $(1, 3)$-DS of the network (same network example presented in [3]): (a) Max-Min, and (b) DKR (*assuming maximum ID nodes are elected, rather than the default minimum ID*)

### 5.2.3 Analysis of DKR

To prove the correctness of DKR, we have to show that it is *safe* (i.e., the algorithm computes a $(k, r)$-DS of the network), and that it is *live* (i.e., it completes within a finite period of time).

**Lemma 1.** Phase one *of DKR has time complexity of* $O(n\delta r)$, *where* $n$ *is the number of nodes in the network,* $\delta$ *is the largest node degree, and* $r$ *is the distance parameter.*

*Proof.* During each round, nodes send messages to all their one-hop neighbors. *Phase one* takes $r$ rounds. Assuming a network of $n$ nodes, and that nodes have at most $\delta$ links, *phase one* of DKR requires $O(n\delta r)$ messages to complete. Therefore, the time complexity of *phase one* is $O(n\delta r)$. □

**Lemma 2.** *After* $r$ *rounds of* **successful** *transmission of message* $\mathbf{m}$, *the message is propagated up to* $r$ *hops away from the originating node.*

*Proof.* This can be proved by induction on the distance $d$ from the originating node. The base case is when $d = 0$, and corresponds to the originating node $n_0$. Now consider a node $v$ at distance $r$ from $n_0$. A neighbor $u$ of node $v$ at distance $r - 1$ received the message. Therefore, node $u$ sends the message to all neighbors, including $v$. Eventually node $v$ receives the message. □

**Theorem 5.** Phase One *of DKR correctly computes a* $(k, r)$*-DS of any arbitrary connected graph* $G = (V, E)$.

*Proof.* We assume that nodes know their one-hop neighbors. The system is either synchronous or asynchronous. In the latter case, rounds are simulated by tagging advertisements with the round number. By Lemma 2, after $r$ rounds a node ID is propagated at most $r$ hops away. Because nodes advertise their $K \leq k$ known smaller IDs, after $r$ rounds every node $n \in V$ learns the $K \leq k$ nodes, $D_n^{'}$, with smaller IDs located within distance $r$. Lets assume that $S$ is the set composed of *proper-elected*, *self-elected*, and unsatisfiable nodes (i.e., a node $i$ is deemed unsatisfiable if $|D_i^{'}| < k$), and that $R$ is the set of satisfiable non-*proper/self-elected* nodes (i.e., $R = V - S$). It follows that the set $D = \{\bigcup_{n \in R} D_n^{'} + S\}$ is a $(k, r)$-DS of $G$. □

**Theorem 6.** *Phase Two of DKR correctly connects dominated nodes to* ***at least*** $k$ *dominating nodes* ***at most*** $r$ *hops away.*

*Proof.* At the beginning of *phase two*, any *dominated* node $i$ advertises its list of elected nodes, $D_i^{'}$, by locally broadcasting the list via an LA message. Any node $n$ with status *dominated*, or *gateway*, upon receiving an LA message from neighbor $m$, changes its status to *dominating* if the advertisement lists node $n$, implying that node $n \in D_m^{'}$. In this case, node $n$ sends an

NA message which is then propagated to all nodes within distance $r$ from node $n$. For every node $k$ in the LA message such that $m \ni D_n^{'}$, a notification is sent to node $k$ if node $n$ is on the path to node $k$ (i.e., for node $n$ the next-hop to node $k$ is known, and it is not $m$). If this is the case, at least one neighbor of node $m$ has a route to node $k$, because nodes are elected based on the advertisements sent by one-hop neighbors during *phase one*. Eventually, the LA message sent by node $m$ reaches all its one-hop neighbors, including the nodes with routes to the nodes elected by node $m$. A Notification for node $n$, when necessary, is issued (initiated or relayed) just once by any node $i$. A *dominated* node relaying a notification changes its status to *gateway*. Once a *Notification* reaches the destination, if the status of the destination is not *dominating*, it changes its status to *dominating*, and advertises itself sending an NA message. If any node $i$ relaying an NA messages currently has fewer than $k$ validated entries in $D_i^{'}$, then an entry with the validated node is inserted in $D_i^{'}$. After a period of time equal to *Wait Period* (starting from the beginning of *Phase Two*), every node $i$ in the network checks the number of *validated* entries in $D_i^{'}$. If node $i$'s status is *pending dominating*, and it has $l < k$ validated entries in $D_i^{'}$, then node $i$ changes its status to *dominating*, and it sends an NA message. Otherwise, if node $i$'s status is not *dominating*, it sends a *Join* message to all its dominating nodes (validated or not). Any node $n$ receiving a *Join* message, for destination $d$, does not need to relay the message in case node $n$ had already initiated, or relayed, a *Join* or *Notification* message to node $d$ before. After all the *Notification*, or *Join*, messages have reached their destinations, the paths from dominated nodes to their respective dominating nodes are formed by nodes that are either *dominating* or *gateway*. Because all nodes must check their status after a finite period of time (i.e., *Wait Period*), and any non-*dominating* node

$i$ must send *Join* messages to all its *dominating* nodes, $(k, r)$-coverage is guaranteed. □

## 5.3 Performance

Even though KR and DKR compute a $(k, r)$-DS of any arbitrary topology, the topologies considered for simulations are those of wireless networks (i.e., modeled using *unit disk graphs* [13]). Given that the $(k, r)$-DS problem is NP-Complete and that KR is the first known approximation, we use it as a lower bound to assess the performance of the distributed solution.

To focus on the efficiency of the heuristics themselves, we use a customized simulator for ad hoc networks, and assume an ideal MAC protocol with which no collisions can occur. This is the same approach adopted in all prior work [15, 37, 68, 69] to compare the efficacy of heuristics. As discussed previously, DKR works in both synchronous and asynchronous networks. However, for the simulations we assume synchronous networks.

Experiments are repeated for $100$ trials with different network topologies, varying the number of nodes and terrain size. Nodes are randomly placed over the terrain, and connectivity is tested to ensure that the network is connected. The radio range is set to $250m$. The results represent the average over the $100$ different networks . The network size is varied from $100$ nodes to $500$ nodes, with increments of $50$ nodes. For the same number of nodes, we vary the terrain size according to two configurations so that we can test the algorithms for different node density (see Table 5.1 for terrain dimensions). Configuration $1$ has a node density of $50\ \frac{\text{nodes}}{km^2}$, and Configuration $2$ has $100\ \frac{\text{nodes}}{km^2}$.

To give an idea of the characteristics of the networks being evaluated, Table 5.2

**Table 5.1**: Terrain Size (in meters)

| # of nodes | Configuration 1 | Configuration 2 |
|---|---|---|
| 100 | 1414 x 1414 | 1000 x 1000 |
| 150 | 1732 x 1732 | 1225 x 1225 |
| 200 | 2000 x 2000 | 1414 x 1414 |
| 250 | 2236 x 2236 | 1581 x 1581 |
| 300 | 2449 x 2449 | 1732 x 1732 |
| 350 | 2645 x 2645 | 1871 x 1871 |
| 400 | 2828 x 2828 | 2000 x 2000 |
| 450 | 3000 x 3000 | 2121 x 2121 |
| 500 | 3162 x 3162 | 2236 x 2236 |

presents the values for the network diameter (i.e., the largest distance between any pair of nodes), and the average node degree for all network sizes. These results show that as the network size increases, so does the network diameter. But it also shows that, in each configuration, the average node degree is similar for all network sizes. In Configuration $2$, nodes have almost twice as many neighbors compared to the networks from Configuration $1$. On the other hand, the network diameter is smaller for networks in Configuration $2$, because there are in average twice as many nodes spread over the same area compared to Configuration $1$.

Three performance metrics are evaluated:

- *Signaling overhead*, which consists of the total number of control packets exchanged for gathering topology information in a centralized algorithm, or for the execution of the two phases in the distributed algorithm.

- Total number of *cluster-heads* (i.e., *dominating* nodes), which is the number of $(k, r)$-dominating nodes for different configurations.

- *Cluster-head sparseness*, which gives a measure of the efficacy of distributing cluster-

**Table 5.2**: Network Diameter and Node Degree (results represent the average $\pm$ std over $100$ samples)

| | Configuration 1 | | Configuration 2 | |
|---|---|---|---|---|
| # of nodes | Diameter | Degree | Diameter | Degree |
| 100 | $10.4 \pm 0.1$ | $7.8 \pm 0.1$ | $6.2 \pm 0.1$ | $14.9 \pm 0.1$ |
| 150 | $12.8 \pm 0.2$ | $8.1 \pm 0.1$ | $7.9 \pm 0.1$ | $15.5 \pm 0.1$ |
| 200 | $14.7 \pm 0.1$ | $8.2 \pm 0.1$ | $9.2 \pm 0.1$ | $16.2 \pm 0.1$ |
| 250 | $16.5 \pm 0.1$ | $8.3 \pm 0.1$ | $10.2 \pm 0.1$ | $16.5 \pm 0.1$ |
| 300 | $18.1 \pm 0.1$ | $8.3 \pm 0.1$ | $11.3 \pm 0.1$ | $16.7 \pm 0.1$ |
| 350 | $19.4 \pm 0.1$ | $8.5 \pm 0.1$ | $12.2 \pm 0.1$ | $16.9 \pm 0.1$ |
| 400 | $20.9 \pm 0.2$ | $8.7 \pm 0.1$ | $13.0 \pm 0.1$ | $17.1 \pm 0.1$ |
| 450 | $21.9 \pm 0.1$ | $8.7 \pm 0.1$ | $13.8 \pm 0.1$ | $17.2 \pm 0.1$ |
| 500 | $23.4 \pm 0.1$ | $8.7 \pm 0.1$ | $14.6 \pm 0.1$ | $17.3 \pm 0.1$ |

heads over the network. It gives the average number of cluster-heads, within the *distance* parameter, per regular node. Results closer to the *multiple domination* parameter means better distribution of cluster-heads. Ideally, regular cluster nodes should have **exactly** $k$ cluster-heads within distance $r$, but this is not a requirement when solving the $(k, r)$-DS problem (which stipulate *at least* $k$ dominating nodes).

To apply KR, complete topology information must be gathered using reliable flooding of link-state updates. In the distributed algorithm, HELLO messages are used for obtaining the one-hop neighbor information, and control messages are reliably transmitted during the two phases of DKR.

Figure 5.8 presents the *signaling overhead* for Configuration 1 (results for Configuration 2 are omitted because they are similar to Configuration 1). The control overhead incurred by the centralized algorithm is due only to the dissemination of topology information. After the nodes have complete topology information, they can compute clusters locally for any parameters. For the distributed algorithm, the signaling overhead varies mostly due to the dis-

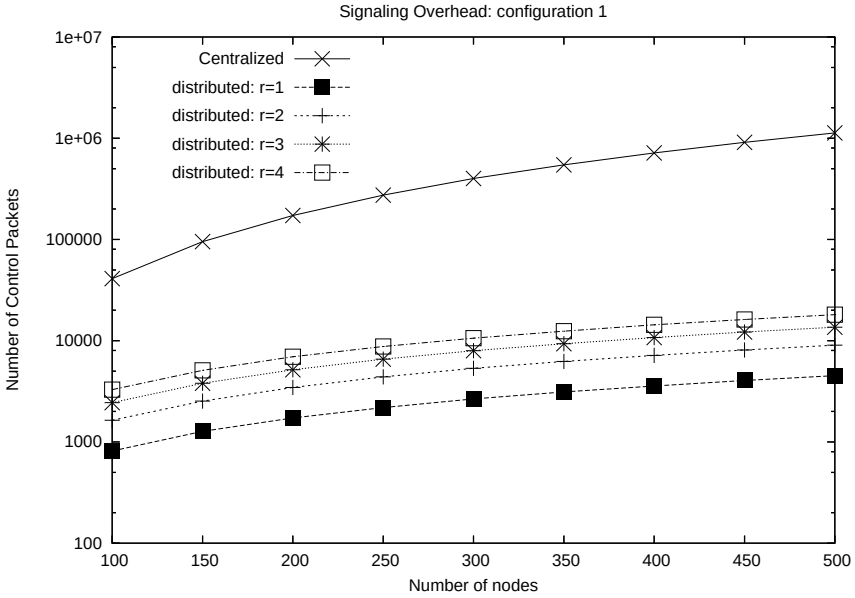



**Figure 5.8**: Centralized versus distributed: *signaling overhead*.

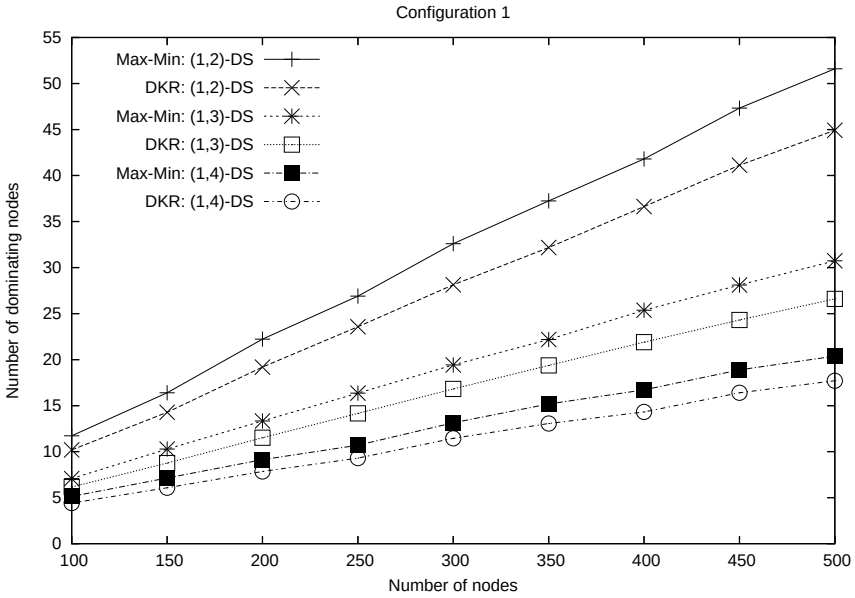



**Figure 5.9**: DKR versus *Max-Min*, Configuration 1: # of *cluster-heads*.

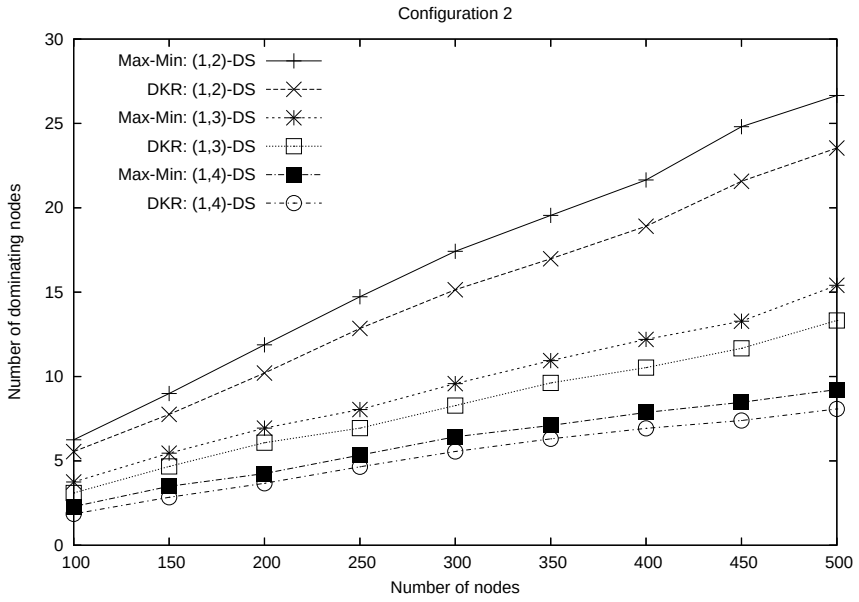



**Figure 5.10**: DKR versus *Max-Min*, Configuration 2: # of *cluster-heads*.

tance parameter. The number of control packets increases as the number of rounds increases; that is, larger the distance parameter, larger the number of advertisements. To show how the signaling overhead varies with the distance parameter, results are presented for parameter $r$ varying from 1 to 4 when computing a $(1, r)$-DS of the network.

Because DKR discards self-elected nodes whenever possible, in the worst case all elected nodes become *cluster-heads*. However, in *Max-Min* [3] all nodes elected at the end of the first $r$ rounds become cluster-heads; but, it is only during the second set of $r$ rounds that some of the elected nodes find out about their *dominating* status. Besides that, in certain scenarios *Max-Min* generates cluster-heads that are on the path between a node and their elected cluster-heads. In that case, only during the *convergecast* (which is used to connect regular nodes to their cluster-heads) that nodes adjust their selections to the closest cluster-head. To show how DKR reduces the number of cluster-heads compared to *Max-Min*, when

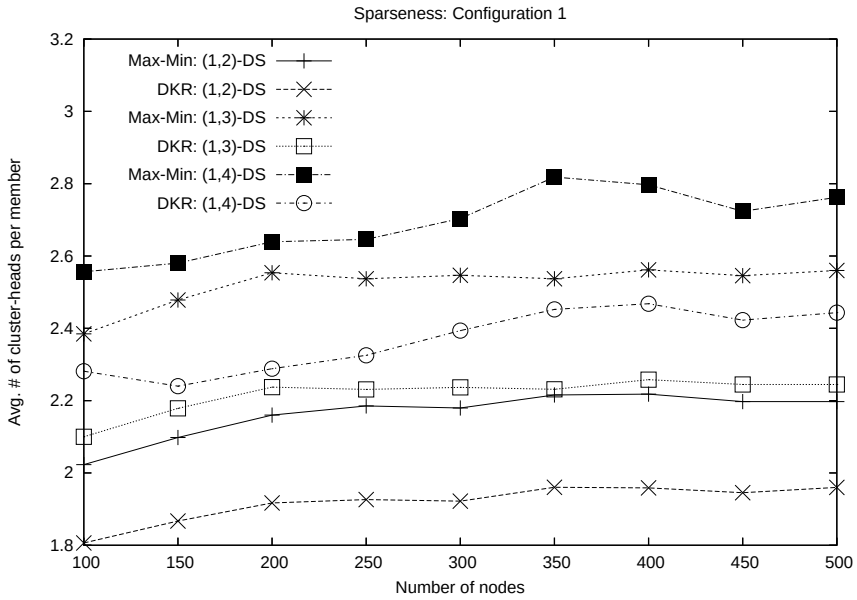



**Figure 5.11**: *DKR* versus *Max-Min*: *cluster-head sparseness*.

computing $(1, r)$-DS (recall that *Max-Min* computes only $r$-hop dominating sets), we present simulations for different values of the distance parameter.

Figure 5.9 presents the results for the total number of cluster-heads for Configuration $1$, varying the distance parameter from $2$ to $4$. And Figure 5.10, presents the results for Configuration $2$. For both configurations DKR always selects fewer cluster-heads compared to *Max-Min*, meaning that usually some *self-elected* nodes are ruled out as cluster-heads.

Figure 5.11 presents the *cluster-head sparseness* results for DKR and *Max-Min* for the networks in Configuration $1$. DKR not only reduces the total number of *cluster-heads*, but it also distribute them more evenly over the network. Recall that, in DKR, some *pending dominating* nodes do not become cluster-heads, and regular nodes join the closest elected nodes.

Figure 5.12 and 5.13 present the results in terms of total number of *cluster-heads* for

Configuration 1 when computing $(k, 2)$-DS, $(k, 3)$-DS, and $(k, 4)$-DS using KR and *DKR*. As expected, *DKR* produces more cluster-heads. For dominating distance two (i.e., $r = 2$), both algorithms behave similarly, presenting a large difference between one dominating node (i.e., $k = 1$) and two-dominating nodes (i.e., $k = 2$). As we increase the distance parameter, fewer nodes are necessary for the dominating set. In KR, each time a dominating node is selected, it spans (i.e., dominates) a larger set of nodes. In *DKR*, nodes with smaller IDs get elected by more nodes farther away as the radius of the election increases. *DKR* produces similar results for $(1, 2)$-DS and $(2, 3)$-DS, because the election of one cluster-head in the two-hop neighborhood of any node increases the chances that, at the end, more elected nodes exist in the three-hop neighborhood of any node. Results for KR computing $(3, 3)$-DS are similar to those for *DKR* computing $(2, 3)$-DS. Similarly, results for KR computing $(3, 4)$-DS are close to those for *DKR* computing $(2, 4)$-DS (Figure 5.13. In both cases, it shows that while the election of nodes is a simple and economic (in terms of overhead) solution, it is shown to be not as efficient as the centralized solution, because the election does not take into account the coverage of nodes when selecting dominating nodes. In case any coverage information should ever be a requirement for the election, this extra information would certainly increase signaling overhead.

Figure 5.14 presents the results in terms of total number of *cluster-heads* for Configuration 2 using KR and *DKR*, when computing $(k, 2)$-DS and $(k, 3)$-DS of the networks. Both approaches produce in average half the number of cluster-heads compared to the results for Configuration 1. This is expected, because in Configuration 2 the networks have smaller diameter, and nodes have almost twice as many neighbors. However, the network diameter

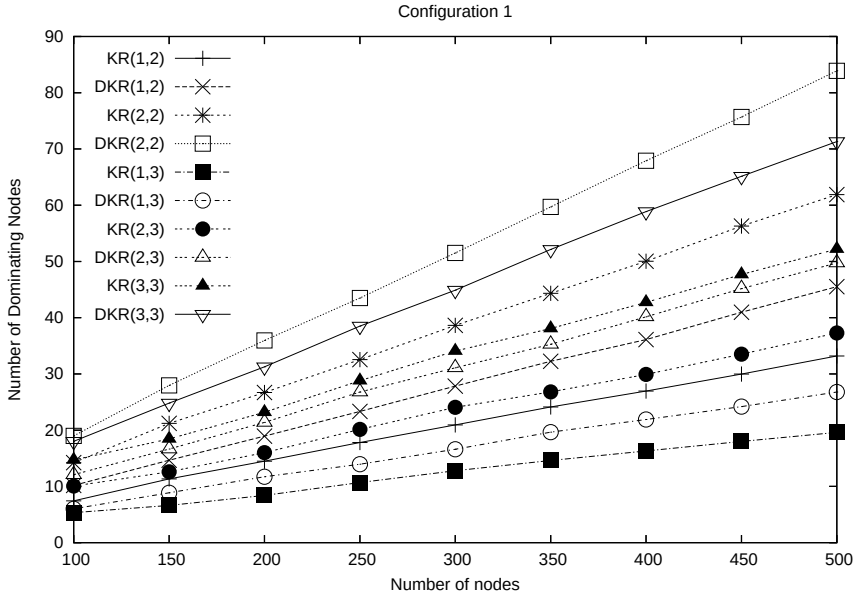



**Figure 5.12**: Conf. 1, computing $(k, 2)$-DS and $(k, 3)$-DS: # of *cluster-heads*.

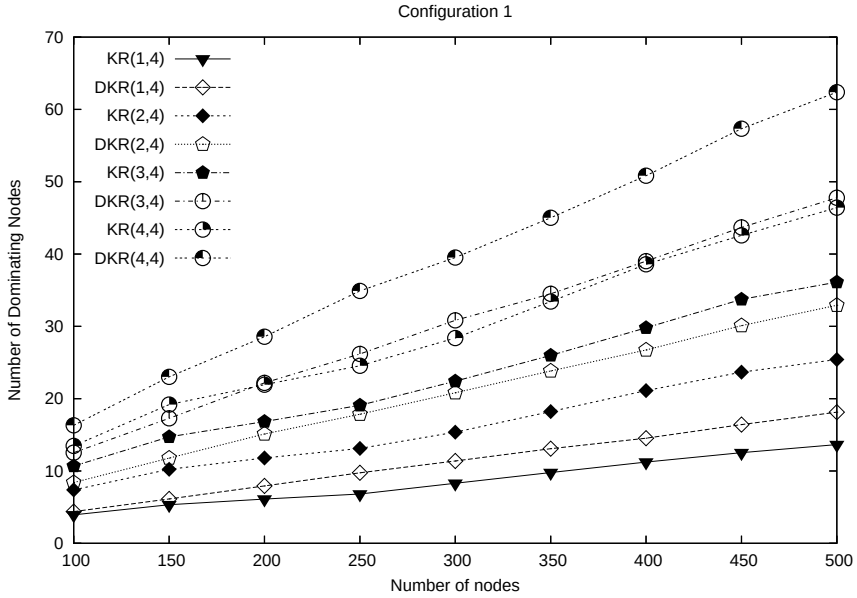



**Figure 5.13**: Conf. 1, computing $(k, 4)$-DS: # of *cluster-heads*.

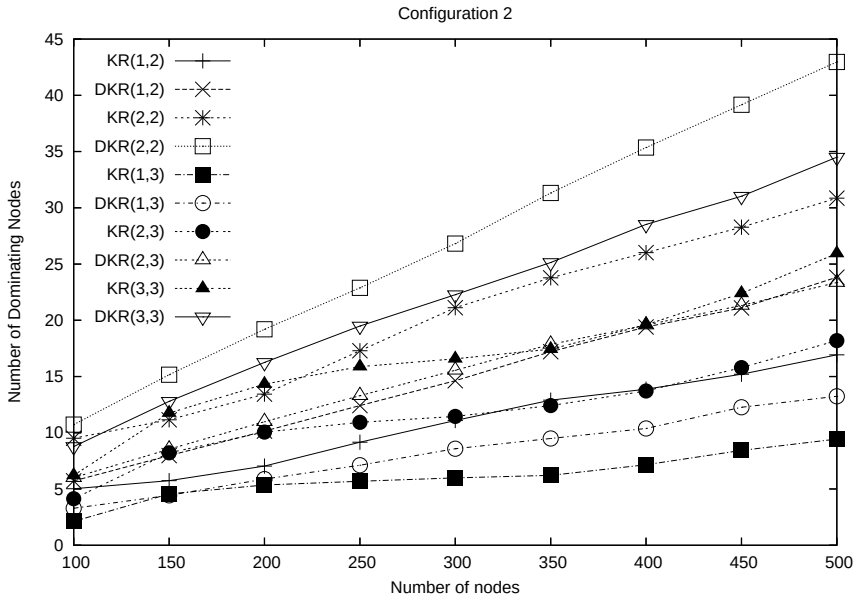



**Figure 5.14**: Conf. 2, computing $(k, 2)$-DS and $(k, 3)$-DS: # of *cluster-heads*.

does not grow as fast as in the networks from Configuration 1. The diameter increases in average less than one unit, as the network size is incremented by 50 nodes (with the exception of the initial jump from approximately 6.2 to 7.9 (see Table 5.2)). Because the network diameter grows slower, so does the cardinality of the DS computed.

Figure 5.15 presents the results when computing $(k, 4)$-DS for the networks in Configuration 2. For networks with 200 to 350 nodes, the distributed algorithm performs well compared to the centralized. Because the distance parameter is larger compared to those from the previous scenarios, and the network diameter does not grow too much for larger networks, it seems that, for relatively small diameter networks, as the radius of the election process increases, much more nodes select the same set of dominating nodes, hence reducing the total number of *cluster-heads*. In addition to that, *self-elected* nodes are more likely to be ruled out as cluster-heads.

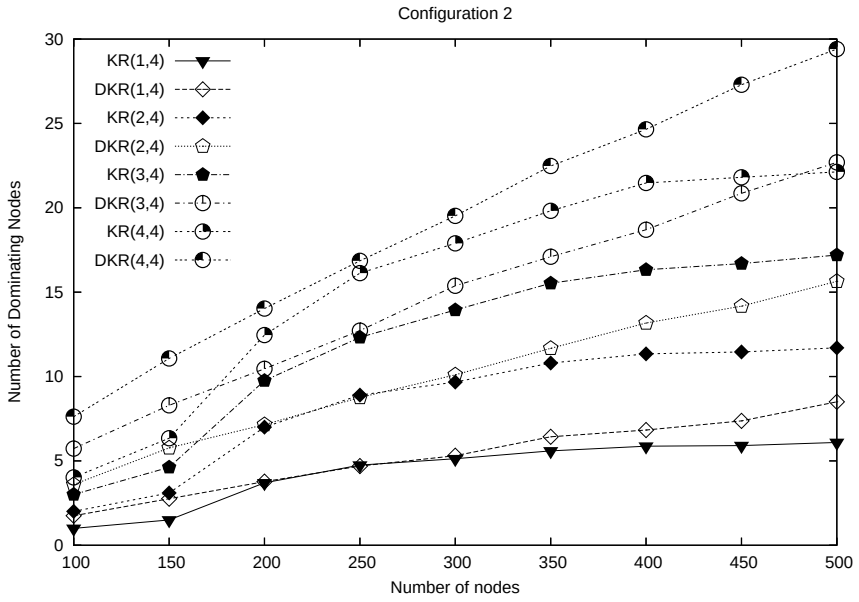



**Figure 5.15**: Conf. 2, computing $(k, 4)$-DS: # of *cluster-heads*.

There is a clear trade-off between efficiency, and communication cost. For MANETs, it pays-off to increase the average number of cluster-heads per cluster, considering that keeping an accurate view of the entire network topology is not possible, and redundancy is desirable.

Figures 5.16 and 5.17 presents the results for the *cluster-head sparseness* metric, when computing $(k, 4)$-DS for the networks in Configuration 1 and 2. Lower values for this metric indicate a better distribution of cluster-heads over the network. Previously, for the same scenarios, it was shown that the networks in Configuration 1 produce more cluster-heads than the networks in Configuration 2. As expected, the *cluster-head sparseness* improves when fewer nodes are selected as cluster-heads. In addition to that, we must compare these results to the corresponding *multiple domination* parameter. That is, as the *cluster-head sparseness* metric approaches the *multiple domination* parameter, better the distribution of cluster-heads.

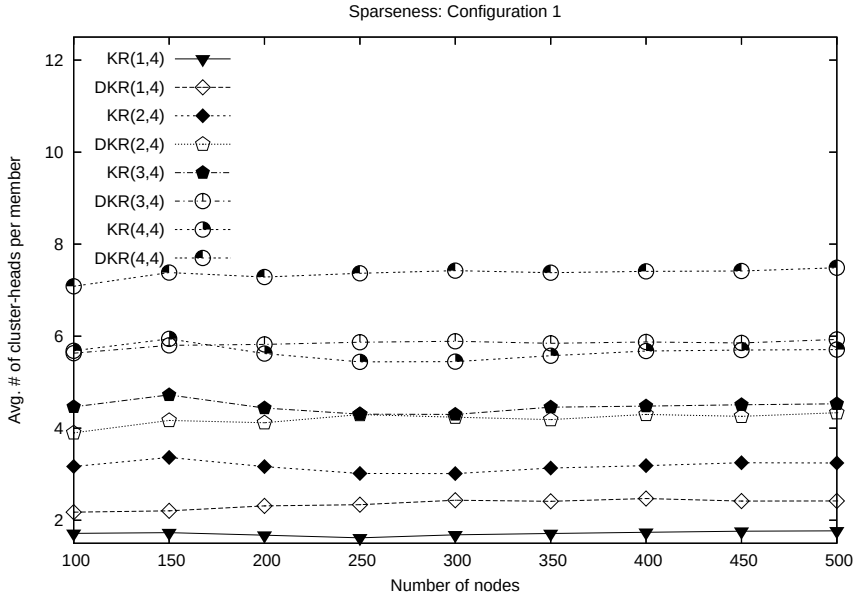



**Figure 5.16**: Conf.1, computing $(k, 4)$-DS: *cluster-head sparseness*.

Ideally, each node should have **exactly** $k$ cluster-heads within distance $r$, but this is not a requirement for solving the $(k, r)$-DS problem, which requires *at least* $k$ cluster-heads.

Because more nodes exist in the $r$-hop neighborhoods of the networks in Configuration 2, the fewer cluster-heads selected are expected to affect a larger subset of nodes compared to Configuration 1. Since KR is sub-optimal, it outperforms DKR in any situation under consideration, presenting results for the *cluster-head sparseness* metric closer to the corresponding *multiple domination* parameter. However, in Configuration 2 (with denser networks, and fewer cluster-heads) the two algorithms present similar results when the *multiple domination* parameter assume values 1 and 2. For larger values of this parameter, the difference increases, because in Configuration 2 the networks have smaller diameter (as shown in Table 5.2).

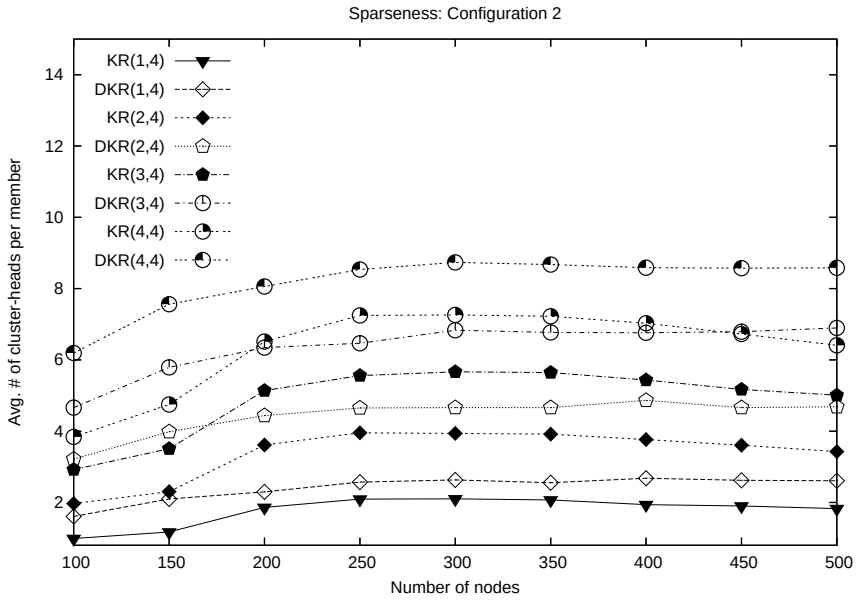



**Figure 5.17**: Conf.2, computing $(k, 4)$-DS: *cluster-head sparseness*.

## 5.4 Conclusions

*Clustering* is the problem of building a hierarchy among nodes, and usually entails the computation of a *dominating set* (DS) of the network. The $(k, r)$*-dominating set* problem, $(k, r)$-DS, is defined as the problem of selecting a minimum cardinality vertex set D (*dominating* nodes, also referred as *cluster-heads*) of a graph $G = (V, E)$, such that every vertex $u$ not in D is at a distance smaller than or equal to $r$ (*distance* domination parameter) from at least $k$ (*multiple* domination parameter) vertices in D. When applying $(k, r)$-DS for clustering, one can define the degree of redundancy (i.e., minimum number of *cluster-heads* per cluster) for clusters of variable radius (maximum distance from regular cluster members to their cluster-heads).

We have presented the first centralized and distributed solutions to the $(k, r)$-DS problem for arbitrary network topologies. The centralized solution, KR, is appropriate for

wired networks, provides an approximation to the optimal solution, and is used as a lower bound when evaluating the performance of the distributed solution. The distributed solution, DKR, computes clusters with variable degree of redundancy, and variable radius. DKR selects *cluster-heads* through an election process. After the election, an approach is used for pruning redundant cluster-heads, and for connecting regular nodes to their cluster-heads.

We have conducted extensive simulations comparing KR against DKR. As expected, KR produces fewer cluster-heads than DKR. However, KR incurs too much *signaling overhead*, making it not appropriate for MANETs. There is a clear trade-off between efficiency (i.e., which approach reduces most the number of *cluster-heads*), and communication cost. While DKR usually produces more cluster-heads per cluster, it does not incur much control overhead.

# Chapter 6

# Multi-Core Multicast Protocol Using (k,r)-DS

Core placement has a direct impact on the performance of multi-core multicast routing protocols. The selection of cores could be further extended to include a minimum number of cores within a maximum distance. In this context, the problem of finding the location of the cores is similar to computing a $(k, r)$-DS of the network. When selecting cores, redundancy is achieved by choosing a value for the parameter $k$ greater than one. At the same time, the distance parameter $r$ allows increasing local availability by reducing the distance to the cores.

We present a novel multi-core multicast protocol named *core hierarchical election for multicasting in ad hoc networks* (*CHEMA*), which uses DKR for the election of cores, and is designed to work in the context of multiple-channel and multiple-interface.

In CHEMA, each node is equipped with two interfaces. One for general communication, and the other for communication among cores and their members. Cores transmit

packets to their members on a specific non-interfering channel via the dedicated interface, and receivers listen in the corresponding channel in the same interface. To reach all members with a single transmission, cores transmit packets with a larger power, such that all member within $r$-hops from the cores can successfully receive the packet. Therefore, all packets transmitted by the cores are expected to be successfully received by the members.

## 6.1 Core Hierarchical Election for Multicasting in Ad Hoc Networks (CHEMA)

In this section we propose *core hierarchical election for multicasting in ad hoc networks* (CHEMA), a new approach for enhancing multicasting in MANETs. CHEMA's main features can be summarized as follows:

- Deploy multiple cores, with DKR as the core selection mechanism. This allows flexibility in terms of redundancy, and a bounded distance to the selected cores. *Core announcements* are used to disseminate core information throughout the network. To reduce overhead, a single announcement aggregates information about all known cores.

- Use the senders-to-all approach. To reduce overhead, the packet header lists which neighbors should relay the multicast packet on a core basis. Instead of sending one packet toward each core, nodes relay the packet whenever they are listed at least once as a next-hop to any core. Before relaying the packet, the header is updated with entries for those cores for which the node is requested to forward the packet.

- Multi-channel and multi-interface: Each node has at least two interfaces. One is dedi-

cated to receiving multicast transmissions from cores, and the other is used for any other transmission. Each core transmits in a channel different than any possible interfering core. That is, through core announcements nodes learn about all cores in the network, and the distance to each one. Using this information, cores select channels so that they do not interfere with other cores.

- Single *shot* approach: Once a multicast packet reaches the core, the packet is transmitted just once via the dedicated interface. In order to reach all receivers, the packet is transmitted with an increased power so that all nodes within $r$ hops from the core can receive it. Because cores use different channels, and an interface is dedicated for receiving packets from the core, receivers should receive all transmissions sent by the core.

Multiple cores are selected via DKR. While cores have not yet been elected, multicast data packets are transmitted via *blind flooding*. After cores are elected, receivers join the nearest core by sending a join message to the core. Nodes aggregate all the fresh core announcements they receive, and broadcast them periodically every *core announcement interval* (which by default is set to be $3s$). Core announcements also include the number of members each core has. To let cores know about any associated members, an explicit *multicast join* message is sent from the receiver to the desired core whenever a node wants to join a multicast group. Note that this is not the same as the association provided by the join messages sent during the execution of DKR, which provides for connectivity from any node to at least $k$ cores within distance $r$.

Senders send multicast packets to all cores with members. Instead of sending one packet per core, the sender broadcasts just one packet with all the information regarding the cores that need to be reached. That is, the packet header includes an entry for every core and the corresponding next hop toward the core (recall that in DKR nodes keep information about which neighbors are used to reach each core). A node receiving the packet for which it is listed as a next hop to any core forwards the packet. Before relaying the packet, entries for which the node is listed as a next-hop are updated with the current information, and any other entries are excluded.

Because multicast packets are broadcast unreliably, a node may retransmit a packet up to N times, unless it receives an implicit acknowledgment. That is, for every multicast packet transmitted the node relaying the packet keeps record of the packet and which neighbors should relay the packet. After a period of time equivalent to an *acknowledgment timeout*, the node checks if it has overheard any of the relayers transmissions. If the node fails to hear any of the relayer's transmission, the node retransmit the packet including only those nodes from which it has not yet heard from. Ideally, the length of an *acknowledgment timeout* should be set dynamically, because it depends on the level of contention, which is higher with a larger number of transmitting nodes.

Nodes are equipped with two radio interfaces. One is used for communication between cores and receivers, and the other is for general communication. More specifically, cores use a dedicated interface for transmitting multicast packets to their members, and the receivers use the same interface to receive packets from their cores. To allow for multiple cores to transmit simultaneously without interference, we assume that each core transmits on

a different channel than any possibly interfering core. Therefore, the dedicated interface is set for transmitting and listening using a specific channel.

The problem of assigning non-interfering channels to cores is similar to the *graph coloring* problem in graph theory. Considering a core with transmission range of $r$, any core within distance $2r$ may interfere (i.e., even though the two cores may be out of range of each other, they may have members within range of both cores). In the context of MANETs, any distributed approximation to the graph coloring problem could be applied for assigning channels to the cores. Instead, we choose to limit the total number of cores in the network to the maximum number of orthogonal channels available for the dedicated interface. Because nodes learn about all cores in the network (through the periodical core announcements), channels are assigned lexicographically.

To reduce delay, and to avoid retransmissions from nodes between the core and members located more than one hop away from the core, cores transmit multicast data packets with a larger power. The transmit power should be set so that the packet can be successfully received by any node up to $r$ hops from the core. Even though this approach increases energy consumption, it is expected to reduce the end-to-end delay and control overhead, because a single transmission from the core is supposed to reach all core members at the same time.

### 6.1.1 Performance

We compare CHEMA against the *protocol for unified multicasting through announcements* (PUMA) [63]. PUMA has been shown to outperform two of the state of the art multicast routing protocols for MANETs (i.e., ODMRP [33] and MAODV [47]). PUMA

**Table 6.1**: Simulation Parameters

| | |
|---|---|
| Simulator | Qualnet™3.5 |
| Simulation time | $350s$ |
| Terrain Size | 1000 X 1000 m |
| Number of nodes | 50 |
| Node placement | Random |
| Mobility | Static |
| Radio Range | $250m$ |
| MAC protocol | 802.11 |
| Channel Capacity | 2 Mbps |
| Data packet size | 512 Bytes |

presents the following characteristics: receiver-oriented; core based (one core per group); mesh-based, providing multiple routes from senders to receivers.

Only the core performs control packet flooding in PUMA. In CHEMA it is the same, but information about multiple cores are aggregated to reduce control overhead (PUMA also applies aggregation when flooding information about multiple groups).

We compared CHEMA against PUMA using the Qualnet™ [1] network simulator. Each simulation was run with four different random-number seeds. Timer values (i.e., *core announcements* in CHEMA, and *multicast announcements* in PUMA) were set to three seconds. Table 6.1 presents details about the simulation parameters.

In CHEMA, cores use a multiple of the regular radio range (i.e., for distance domination $r$, cores have a radio range of $r \cdot 250m$) for the dedicated interface. DKR is executed every $16s$ for core assignment. Because only static topologies are considered, the cores remain the same throughout the simulation.

Four performance metrics are evaluated:

- Packet delivery ratio: The ratio of the data packets delivered to the receivers to those

data packets expected to be delivered (i.e., data packets sent times the number of receivers).

- Average end-to-end delay for data packets, including all possible delays caused by queuing at the interface, retransmission delays, and propagation and transfer times.
- Control overhead: The number of control packets transmitted per data packet delivered.
- Total overhead: The ratio of the total packets transmitted (i.e., control + data) to the data packets delivered.

For the simulation scenario, traffic load is varied across $\{1, 2, 5, 10, 25, 50\}$ packets/s. There are $5$ senders, and $20$ receivers for one multicast group. That is, the number of packets expected to be delivered varies from $20$ packets/s to $1000$ packets/s. Both senders and receivers are chosen randomly among the nodes in the network, and traffic load is equally distributed among all senders.

Even though DKR allows a myriad of scenarios for core selection, we consider just a few configurations for the purpose of simulation. For a $50$-nodes network, at most eight cores are allowed (and there are $8$ orthogonal channels for the dedicated interface). Values $3$ and $4$ are tested for the *distance domination*, and at least one core is selected within the specified distance. These two configurations are presented in the graphs as CHEMA $(1, 3)$-DS, and CHEMA $(1, 4)$-DS, respectively. For the networks considered, three cores are elected in average in the first configuration, and four cores in the second configuration.

Figure 6.1 presents the results for packet delivery ratio. CHEMA delivers almost $100\%$ of the data packets for all trafic loads considered. But PUMA cannot deliver more than

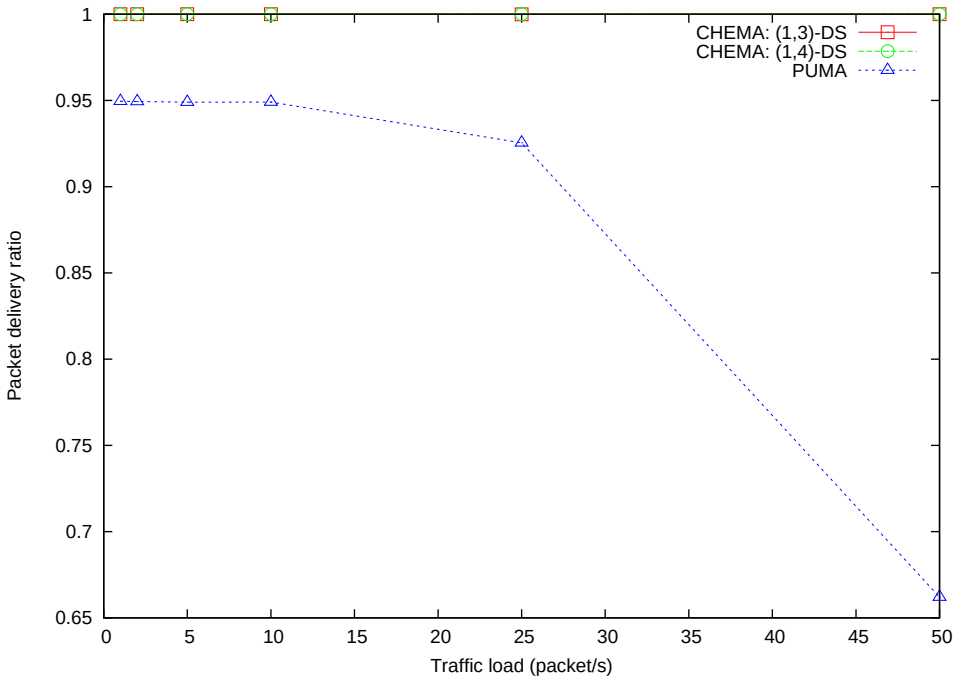



**Figure 6.1**: Packet delivery ratio

$70\%$ of packets for traffic load of $50$ packets/s, due to increasing contention and collision of packets. On the other hand, mainly because CHEMA applies the *one shot* approach and the non-interfering channels for cores, once packets are transmitted by the core the packets are successfully received by the receivers.

For flows of up to $10$ packets/s both protocols present similar results for the end-to-end delay (Figure 6.2). While CHEMA has a small increase in terms of end-to-end delay for flows larger than $10$ packets/s, PUMA experiences an exponential increase in average end-to-end delay. These results, together with the delivery ratio, indicate that CHEMA not only delivers more packets but does so incurring smaller end-to-end delays. This shows that it pays off sending packets to multiple cores and using a single transmission per packet from the cores

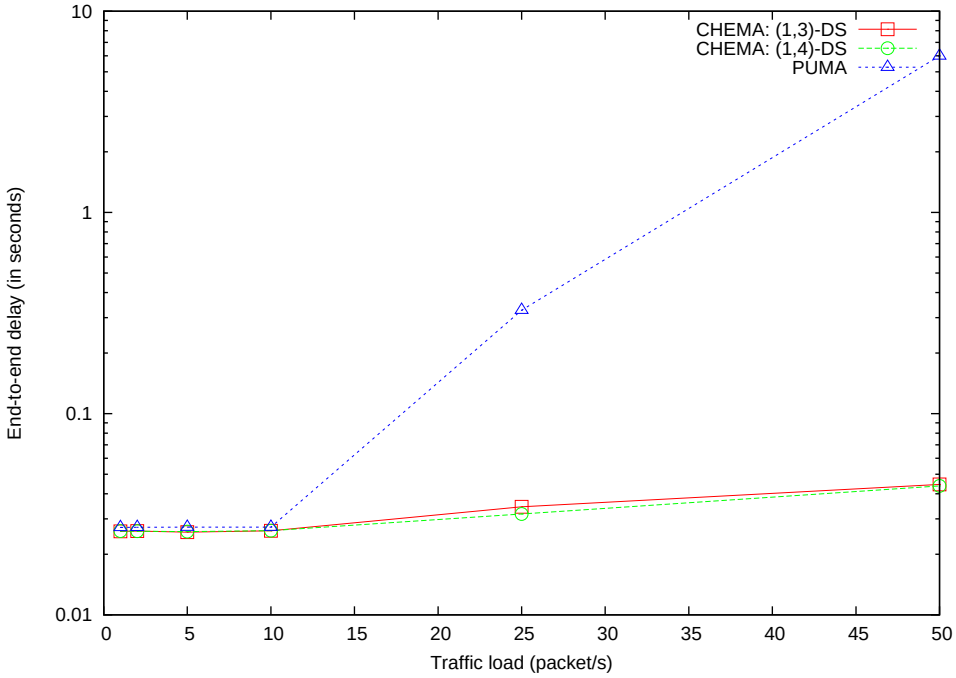



**Figure 6.2**: End-to-end delay

to their members.

Even though CHEMA sends more control packets (mainly due to the election process) compared to PUMA, both protocols present similar control overhead because CHEMA delivers more packets (as shown in Figure 6.3). However, in terms of total overhead CHEMA incurs less than half total overhead compared to PUMA (Figure 6.4). CHEMA requires fewer transmissions for every data packet delivered, specially because once the packets reach the targeted core it takes just one transmission per data packet to reach all core's receivers.

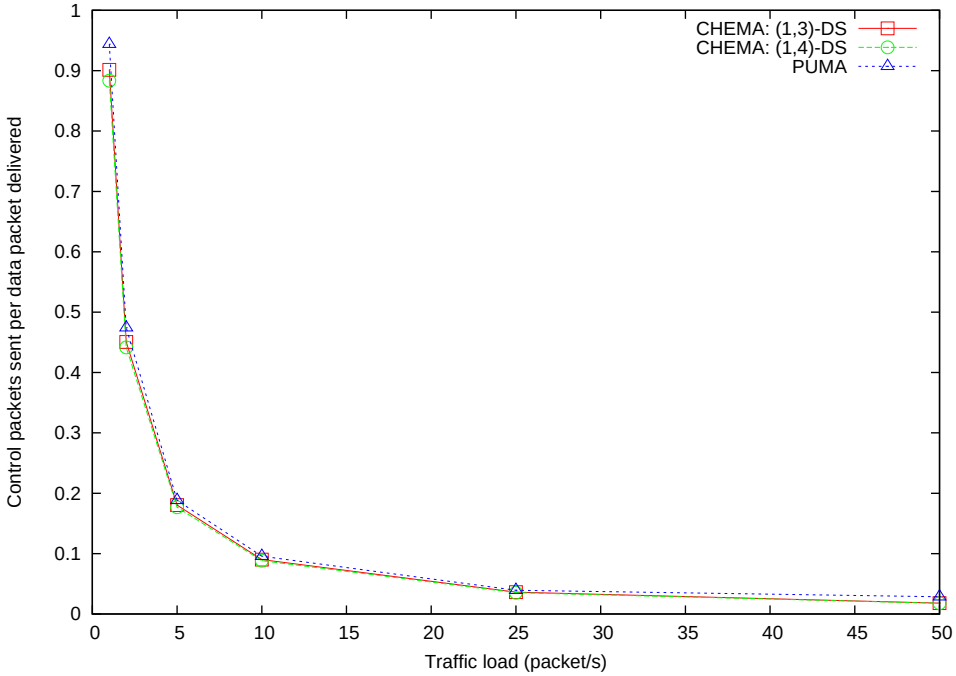



**Figure 6.3**: Control overhead

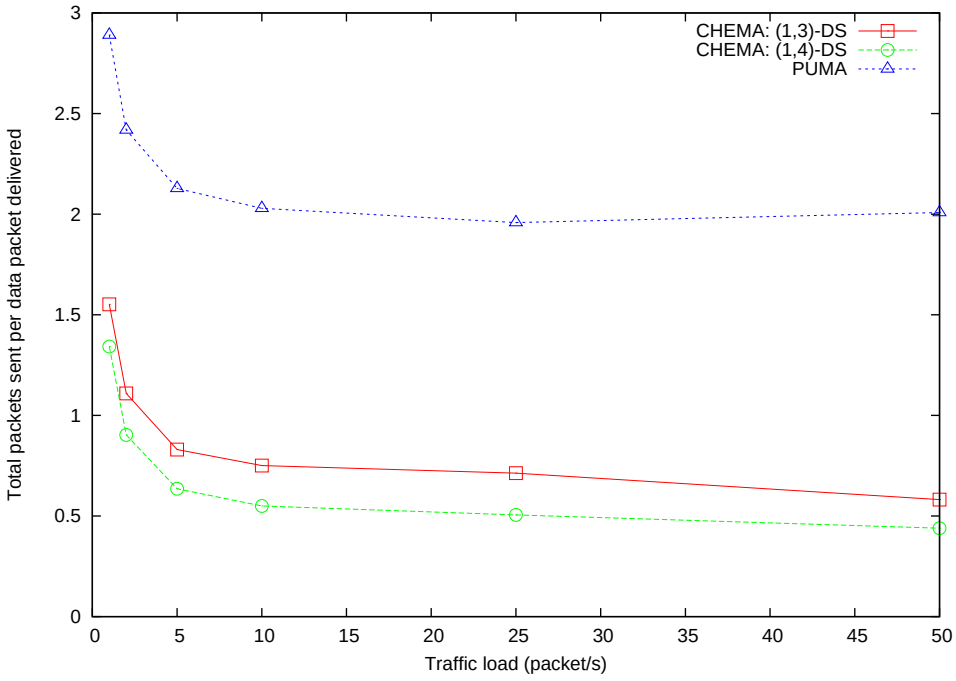



**Figure 6.4**: Total overhead

## 6.2 Conclusion

We proposed a novel multicast protocol named *core hierarchical election for multicasting in ad hoc networks* (*CHEMA*), which is designed to work in the context of multiple-channel and multiple-interface. CHEMA applies DKR for core election, with a dedicated interface using non-interfering channel for communication between cores and any multicast member. Because cores use a larger radio range for the dedicated interface, it requires just one transmission per data packet for any core member to receive the packet.

CHEMA is compared against the *protocol for unified multicasting through announcements* (PUMA), which is one of the best performing multicast routing protocols for MANETs. CHEMA is shown to outperform PUMA in all aspects. CHEMA delivers more packets, incurs small end-to-end delays, and drastically reduces the total control overhead.

# Chapter 7

# Conclusion and Future Work

## 7.1 Conclusion

In this thesis, we present applications of *domination in graphs* in the context of mobile ad hoc networks (MANETs). In ad hoc networks, nodes must coordinate among themselves without resorting to any pre-existing network infrastructure. Coordination of nodes relies on broadcasting of signaling messages, sometimes resulting in some form of organization of nodes (e.g., clustering or topology management). To improve broadcasting and hierarchical organization in MANETs, we explore concepts from *domination in graphs*, because of their similarities to the problems we want to address.

We present a novel algorithm for computing a dynamic source-based dominating set of the network, which is used for improving broadcast operations in MANETs. The solution is shown to reduce the number of broadcast transmissions necessary to flood the network. When the algorithm is applied to the route discovery process of an on demand routing protocol, it is

shown to improve the performance of the protocol in all aspects considered.

Then we present a particular configuration of *connected dominating sets* for enhancing the route discovery process in on demand routing protocols. When a broadcast protocol based on neighbor information is used, it is possible to maintain fresh routes to all nodes within two hops, because every node has the two-hop neighborhood information. In this case, we show that using a connected dominating set with the property that nodes are at most two-hops from a dominating node (i.e., a *two-hop connected dominating set* (TCDS)) is enough for the route discovery process. The proposed algorithm is shown to outperform the best existing heuristics presented in the literature. When integrated into the route discovery process of an on demand routing protocol, in addition to some enhancements, the new protocol is shown to outperform other protocols in all aspects considered.

Motivated by those two successful applications of dominating sets for enhancing broadcast operations, we present a framework based on dominating sets for building flexible hierarchies with the support for fault-tolerant applications. The framework allows building structures that cover a node in the network with a minimum number of dominating nodes and a maximum distance to the dominating nodes. Using this framework, it is possible to build a hierarchical structure such that every node in the network is covered by at least $k$ leaders at most $d$ hops distant. A structure like this could be used to support operations that require increased redundancy, and an adjustable degree of availability. We present a centralized and a distributed solution to this problem. The centralized solution provides an approximation to the optimum solution, which is known to be NP-Complete, and serves as a lower bound when evaluating the distributed solution. The distributed solution is applicable to MANETs,

because it relies on partial topology information.

To show an application for the proposed framework, we present a novel multi-core multicast protocol for MANETs using the framework in the core-election process. The dominating set computed via the distributed algorithm is used for assigning cores. The distributed algorithm is shown to perform well for electing cores, and the new multicast protocol is shown to outperform one of the best known multicast protocols presented in the literature.

## 7.2 Future Work

As shown previously, many approaches for improving broadcasting in MANETs require information about the local neighborhood (e.g., the two-hop neighborhood). It will be interesting to evaluate such approaches over a MAC protocol that exchanges the neighbor and forwarder information that we assume is exchanged as part of the routing protocol itself. This is expected to help maintaining an accurate view of the neighborhood, and improve broadcasting.

EDP is the first heuristic to incorporate the second-to-previous forwarder list in the pruning process. It will be interesting to analyze the impact of extending the history of previous forwarder list when pruning nodes; that is, take into account the third-to-previous forwarder list, the fourth-to-previous forwarder list, and so on.

THP was designed to improve the route discovery process of on-demand routing protocols by computing a TCDS of the network. Instead of just taking into account the coverage of nodes during the pruning process, it will be interesting to investigate how to explore alternate paths between any pair of nodes while computing a TCDS of the network.

All broadcasting mechanisms based on pruning could use different constraints (e.g., energy consumption, interference, and load balancing) when deciding the set of forwarding nodes. It will be interesting to investigate how different constraints relate to each other, and how they compare in terms of efficiency and reliability for the broadcasting of signaling and data packets. For example, “is it possible to design energy-efficient and reliable broadcasting mechanisms?”.

DKR does not employ any load balancing when electing dominating nodes. It will be interesting to investigate solutions to incorporate load balancing when building a $(k, r)$-DS of the network. Investigate how often nodes should trade positions as dominating nodes, and how to better distribute the client load among the elected nodes.

It will be interesting to change DKR to operate in a totally asynchronous mode. That is, operate without any concept of round. Because new nodes can join the network at any time, and current nodes may leave, it would be better to have an election scheme that tolerates transient modifications to the topology, but also guarantees the selection of a not so large set of dominating nodes.

# Bibliography


[1] Scalable network technologies, 2003. Qualnet simulator. http://www.scalable-networks.com.

[2] Ian F. Akyildiz, Su Weilian, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, August 2002.

[3] A. D. Amis, R. Prakash, T. H. P. Vuong, and D.T. Huynh. Max-min d-cluster formation in wireless ad hoc networks. In *IEEE INFOCOM*, pages 32–41, March 2000.

[4] D. J. Baker and A. Ephremides. The architectural organization of a mobile radio network wia a distributed algorithm. *IEEE Transactions on Communications*, 29(11):1694–1701, November 1981.

[5] Tony Ballardie, Paul Francis, and Jon Crowcroft. Core based trees (cbt). In *SIGCOMM '93: Conference proceedings on communications architectures, protocols and applications*, pages 85–95. ACM Press, 1993.

[6] S. Bandyopadhyay and E. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *IEEE INFOCOM*, pages 1713–1723, 2003.

[7] Lichun Bao and J. J. Garcia-Luna-Aceves. Transmission scheduling in ad hoc networks with directional antennas. In *MobiCom '02: Proceedings of the 8th annual international conference on mobile computing and networking*, pages 48–58. ACM Press, 2002.

[8] Lichun Bao and J. J. Garcia-Luna-Aceves. Topology management in ad hoc networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on mobile ad hoc networking & computing*, pages 129–140. ACM Press, 2003.

[9] Marcelo M. Carvalho and J. J. Garcia-Luna-Aceves. A scalable model for channel access protocols in multihop ad hoc networks. In *MobiCom '04: Proceedings of the 10th annual international conference on mobile computing and networking*, pages 330–344. ACM Press, 2004.

[10] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *MobiCom '01: Proceedings of the 7th annual international conference on mobile computing and networking*, pages 85–96. ACM Press, 2001.

[11] G. Chen, F. Nocetti, J. Gonzalez, and I. Stojmenovic. Connectivity based k-hop clustering in wireless networks. In *HICSS '02: Proceedings of the 35th annual Hawaii international conference on system sciences*, pages 2450–2459. IEEE Computer Society, 2002.

[12] Y. P. Chen, A. L. Liestman, and J. Liu. *Ad Hoc and Sensor Networks*, chapter Clustering algorithms for ad hoc wireless networks. Nova Science Publisher, 2004.

[13] B.N. Clark, C.J. Colbourn, and D.S. Johnson. Unit disk graphs. *Discrete Math*, 86:165–177, 1990.

[14] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *Proceedings of the IEEE international multi topic conference*, pages 62–68, December 2001.

[15] F. Dai and J. Wu. Distributed dominant pruning in ad hoc networks. In *Proceedings of the IEEE international conference on communications*, pages 353–357, May 2003.

[16] K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[17] Eli Gafni. Round-by-round fault detectors (extended abstract): unifying synchrony and asynchrony. In *PODC '98: Proceedings of the seventeenth annual ACM symposium on principles of distributed computing*, pages 143–152. ACM Press, 1998.

[18] J. J. Garcia-Luna-Aceves and Marcelo Spohn. Source-tree routing in wireless networks. In *Proceedings of the IEEE international conference on network protocols*, pages 273–282, 1999.

[19] M. R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman, San Francisco, 1978.

[20] M. Gerla, Jin Kwon Taek, and G. Pei. On-demand routing in large ad hoc wireless networks with passive clustering. In *Proceedings of the IEEE wireless communications and networking conference*, pages 100–105, September 2000.

[21] Mario Gerla and J. Tsai. Multicluster, mobile, multimedia radio network. *ACM Baltzer Journal of Wireless Networks*, 1(3):255–265, September 1995.

[22] Sudipto Guha and Samir Khuller. Approximation algorithms for connected dominating sets. In *European symposium on algorithms*, pages 179–193, 1996.

[23] Z. Haas. A new routing protocol for the reconfigurable wireless network. In *Proceedings of the IEEE international conference on universal personal communications*, pages 562–566, October 1997.

[24] Teresa W. Haynes, Stephen T. Hedetniemi, and Peter J. Slater, editors. *Fundamentals of Domination in Graphs*. Marcel Dekker, Inc., 1998.

[25] W. Heinzelman. *Application-Specific Protocol Architectures for Wireless Networks*. PhD thesis, Massachusetts Institute of Technology, 2000.

[26] Michael A. Henning, Ortrud R. Oellermann, and Henda C. Swart. The diversity of domination. *Discrete Mathematics*, 161(1-3):161–173, December 1996.

[27] Jeffrey Hightower and Gaettano Borriello. A survey and taxonomy of location systems for ubiquitous computing. Technical Report UW-CSE 01-08-03, University of Washigton, Computer Science and Engineering, 2001.

[28] L. Hu. Topology control for multihop packet radio networks. *IEEE Transactions on Communications*, 41(10):1474–81, October 1993.

[29] Zhuochuan Huang and Chien-Chung Shen. A comparison study of omnidirectional and directional MAC protocols for ad hoc networks. In *Proceedings of the IEEE global telecommunications conference*, pages 57–61, November 2002.

[30] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, volume 353, pages 153–179. Kluwer Academic Publishers, 1996.

[31] Dipti Joshi, Sridhar Radhakrishnan, and Chandrasekheran Narayanan. A fast algorithm for generalized network location problems. In *Proceedings of the ACM/SIGAPP symposium on applied computing*, pages 701–8, 1993.

[32] Leonard Kleinrock and Farouk Kamoun. Optimal clustering structures for hierarchical topological design of large computer networks. *Networks*, 10:221–248, 1980.

[33] Sung-Ju Lee, Mario Gerla, and Ching-Chuan Chiang. On-demand multicast routing protocol. In *Proceedings of the IEEE wireless communications and networking conference*, pages 1298–1302, 1999.

[34] Xiang-Yang Li. *Ad Hoc Networking*, chapter Topology Control in Wireless Ad Hoc Networks. IEEE Press, 2003.

[35] Ben Liang and Zygmunt J. Haas. Virtual backbone generation and maintenance in ad hoc network mobility management. In *IEEE INFOCOM*, pages 1293–1302, 2000.

[36] H. Lim and C. Kim. Flooding in wireless ad hoc networks. *Computer Communications*, 24(3–4):353–363, February 2001.

[37] Wei Lou and Jie Wu. On reducing broadcast redundancy in ad hoc wireless networks. *IEEE Transactions on Mobile Computing*, 1(2):111–122, April-June 2002.

[38] Tamas Lukovszki. *New Results on Geometric Spanners and Their Applications*. PhD thesis, University of Paderborn, 1999.

[39] M. Mosko and J. J. Garcia-Luna-Aceves. A self-correcting neighbor protocol for mobile ad hoc wireless networks. In *Proceedings of the IEEE international conference on computer communications and networks*, pages 556–560, 2002.

[40] S. Narayanaswamy, V. Kawadia, R. Sreenivas, and P. Kumar. Power control in ad-hoc networks: Theory, architecture, algorithm and implementation of the compow protocol. In *Proceedings of the European wireless conference*, pages 156–162, 2002.

[41] Katia Obraczka, Kumar Viswanath, and Gene Tsudik. Flooding for reliable multicast in multi-hop ad hoc networks. *Wireless Networks*, 7(6):627–634, 2001.

[42] R. Ogier, F. Templin, and M. Lewis. Request for comments 3687: Topology dissemination based on reverse-path forwarding, February 2004.

[43] G. Pei, Mario Gerla, X. Hong, and C. Chiang. A wireless hierarchical routing protocol with group mobility. In *Proceedings of the IEEE wireless communications and networking conference*, volume 3, pages 1538–1542, 1999.

[44] Guangyu Pei, Mario Gerla, and Xiaoyan Hong. Lanmar: landmark routing for large scale wireless ad hoc networks with group mobility. In *MobiHoc '00: Proceedings of the 1st ACM international symposium on mobile ad hoc networking & computing*, pages 11–18. IEEE Press, 2000.

[45] C. Perkins. Ad-hoc on-demand distance vector routing. In *Proceedings of the IEEE workshop on mobile computing systems and applications*, pages 90–100, 1999.

[46] Charles Perkins, Elizabeth Royer, Samir R. Das, and Mahesh K. Marina. Performance comparison of two on-demand routing protocols for ad hoc networks. *IEEE Personal Communications*, 8(1):16–28, February 2001.

[47] Elizabeth M. Royer and Charles E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on mobile computing and networking*, pages 207–218. ACM Press, 1999.

[48] Adrian Segall. Distributed network protocols. *IEEE Transactions on Information Theory*, 29(1):23–35, January 1983.

[49] Nachum Shacham and Jil Westcott. Future directions in packet radio architectures and protocols. *Proceedings of the IEEE*, 75(1):83–99, 1987.

[50] Marco A. Spohn and J. J. Garcia-Luna-Aceves. Enhanced dominant pruning applied to the route discovery process of on-demand routing protocols. In *Proceedings of the 12th IEEE international conference on computer communications and networks*, pages 497–502, October 2003.

[51] Marco A. Spohn and J. J. Garcia-Luna-Aceves. Enhancing broadcast operations in ad hoc networks with two-hop connected dominating sets. In *Proceedings of the IEEE international conference on mobile ad-hoc and sensor systems*, pages 543–545, October 2004.

[52] Marco A. Spohn and J. J. Garcia-Luna-Aceves. Enhancing the route discovery process of on-demand routing in networks with directional antennas. In *Lecture notes in computer science*, volume 3042, pages 1000–1011. Springer-Verlag GmbH, January 2004.

[53] Marco A. Spohn and J. J. Garcia-Luna-Aceves. Improving broadcast operations in ad hoc networks using two-hop connected dominating sets. In *Proceedings of the IEEE global telecommunications conference workshops*, pages 147–152, November 2004.

[54] Marco A. Spohn and J. J. Garcia-Luna-Aceves. Improving route discovery in on-demand routing protocols using two-hop connected dominating sets. *Elsevier Ad Hoc Networks*, 2005. Accepted for publication.

[55] Marco A. Spohn and J. J. Garcia-Luna-Aceves. Improving the efficiency and reliability of the route discovery process in on-demand routing protocols. In *Proceedings of the IEEE wireless communications and networking conference*, pages 2008–2013, March 2005.

[56] Ivan Stojmenovic, Mahtab Seddigh, and Jovisa Zunic. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(1):14–25, 2002.

[57] John Sucec and Ivan Marsic. An efficient distributed network-wide broadcast algorithm for mobile ad hoc networks. Technical Report CAIP-248, Rutgers University, September 2000.

[58] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, 2003.

[59] F. Tobagi and L. Kleinrock. Packet switching in radio channels. *Transaction on Communications*, 23(12):1400–16, 1975.

[60] Godfried T. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.

[61] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless Networks*, 8:153–167, 2002.

[62] P. F. Tsuchiya. The landmark hierarchy: a new hierarchy for routing in very large networks. In *SIGCOMM '88: Symposium proceedings on communications architectures and protocols*, pages 35–42. ACM Press, 1988.

[63] Ravindra Vaishampayan and J. J. Garcia-Luna-Aceves. Efficient and robust multicast routing in mobile ad hoc networks. In *Proceedings of the IEEE international conference on mobile ad-hoc and sensor systems*, pages 304–313, October 2004.

[64] Peng-Jun Wan, Khaled M. Alzoubi, and Ophir Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. In *IEEE INFOCOM*, pages 1597–1604, June 2002.

[65] Y. Wang and J. J. Garcia-Luna-Aceves. Spatial reuse and collision avoidance in ad hoc networks with directional antennas. In *Proceedings of the IEEE global telecommunications conference*, pages 112–116, November 2002.

[66] Y. Wang and J. J. Garcia-Luna-Aceves. Collision avoidance in single-channel ad hoc networks using directional antennas. In *Proceedings of the IEEE international conference on distributed computing systems*, pages 640 – 649, May 2003.

[67] Brad Williams and Tracy Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on mobile ad hoc networking & computing*, pages 194–205. ACM Press, 2002.

[68] Jie Wu and Fei Dai. Broadcasting in ad hoc networks based on self-pruning. In *IEEE INFOCOM*, pages 2240–2250, 2003.

[69] Jie Wu and Fei Dai. A generic distributed broadcast scheme in ad hoc wireless networks. In *Proceedings of the IEEE international conference on distributed computing systems*, pages 460–467, May 2003.

[70] Jie Wu and Fei Dai. Mobility management and its applications in efficient broadcasting in mobile ad hoc networks. In *IEEE INFOCOM*, pages 339–350, 2004.

[71] Jie Wu and Fei Dai. Mobility-sensitive topology control in mobile ad hoc networks. In *Proceedings of the IEEE parallel and distributed processing symposium*, page 28, 2004.

[72] Jie Wu and Hailan Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *DIALM '99: Proceedings of the 3rd international workshop on discrete algorithms and methods for mobile computing and communications*, pages 7–14. ACM Press, 1999.

[73] Ossama Younis and Sonia Fahmy. Heed: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *Transactions on Mobile Computing*, 3(4):366–379, 2004.

[74] Daniel Zappala, Aaron Fabbri, and Virginia Lo. An evaluation of shared multicast trees with multiple cores. *Telecommunication Systems*, 19(3-4):461–479, March - April 2002.